University of Alberta

Handling Uncertainty When You're Handling Uncertainty: Model Selection and Error Bars for Belief Networks

by

Timothy Karl Van Allen ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2000

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-59891-8

Canada

University of Alberta

Library Release Form

Name of Author: Timothy Karl Van Allen

Title of Thesis: Handling Uncertainty When You're Handling Uncertainty: Model Selection and Error-Bars for Belief Networks

Degree: Master of Science

Year this Degree Granted: 2000

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Timothy Karl Van Allen
111 RH Michener Park
Edmonton, AB, T6H 4M4
Canada

Date: Sept. 13, 2000

Abstract

Belief networks are a common way of handling uncertainty in AI. A belief network represents the joint distribution of a set of random variables. When network parameters are estimated from a sample, the parameter values are also random variables whose distribution is given by the sampling distribution of the true model (Frequentist perspective) or the posterior distribution over the parameter space (Bayesian perspective). The uncertainty in parameter values has implications for both inference and learning. In learning network structure from data, a fundamental issue is how to handle the bias-variance trade-off — increasing model complexity decreases bias but increases the variance in parameter values. We compare model selection criteria for handling the bias-variance trade-off in structure learning, on theoretical and empirical grounds. We also look at the issue of the uncertainty in belief network inference. Once constructed, belief networks are typically used to answer queries about marginalizations or conditionalizations of the full distribution. Inferences based on network parameters inherit the uncertainty in those parameters. We present a method for computing approximate error-bars around belief network inferences. An efficient algorithm is given, and empirical evidence of its efficacy is presented.

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Handling Uncertainty when You're Handling Uncertainty: Model Selection and Error-Bars for Belief Networks** submitted by Timothy Karl Van Allen in partial fulfillment of the requirements for the degree of **Master of Science**.

Russell Greiner
Supervisor

Terrence Caelli
Co-Supervisor

Peter Hooper
External Examiner

Date: Sept. 13, 2000

Handling Uncertainty When You're Handling Uncertainty: Model Selection and Error Bars for Belief Networks

Timothy Karl Van Allen

Acknowledgements

I would like to acknowledge the assistance of my supervisor Russ Greiner of the Department of Computing Science, and Peter Hooper of the Department of Mathematics (Statistics Center) in preparing the content of this thesis. In addition, I would like to acknowledge the financial support of the National Science and Engineering Research Council of Canada, and the University of Alberta, Department of Computing Science.

Contents

1 Introduction 1

2 Preliminaries 3
- 2.1 Related Work . . . 3
  - 2.1.1 Belief Networks . . . 3
  - 2.1.2 Model Selection Criteria . . . 3
  - 2.1.3 Error-Bars . . . 4
- 2.2 Notation . . . 4

3 Belief Networks 6
- 3.1 Representation . . . 6
- 3.2 Parameter Estimation . . . 7

4 Comparing Model Selection Criteria 9
- 4.1 Theoretical Background . . . 9
  - 4.1.1 The Bias-Variance Trade-off . . . 9
  - 4.1.2 Error Functions . . . 10
  - 4.1.3 Complexity Penalty Criteria . . . 11
  - 4.1.4 Validation Criteria . . . 11
  - 4.1.5 Bootstrap Criteria . . . 12
  - 4.1.6 A Bayesian Approach? . . . 12
  - 4.1.7 KL-Divergence and Parameter Estimation . . . 14
- 4.2 Empirical Study . . . 14
  - 4.2.1 Criteria . . . 15
  - 4.2.2 True Models . . . 15
  - 4.2.3 Hypotheses . . . 15
  - 4.2.4 Experimental Design . . . 15
- 4.3 Results . . . 16
- 4.4 Discussion . . . 22
  - 4.4.1 Future Work . . . 23

5 Error-Bars for Inference 24
- 5.1 Error-Bars . . . 24
- 5.2 Derivation . . . 24
  - 5.2.1 Handling Parameter Dependencies . . . 25
  - 5.2.2 Partial Derivatives of Belief Net Queries . . . 26
  - 5.2.3 Efficient Computation of Partial Derivatives . . . 26
  - 5.2.4 Algorithm . . . 29
- 5.3 Empirical Study . . . 30
  - 5.3.1 Experimental Design . . . 30
  - 5.3.2 Results for the Diamond Graph . . . 31
  - 5.3.3 Results for Alarm . . . 33
  - 5.3.4 Results for Random Networks . . . 33

5.3.5 Discussion 35

6 Conclusion **36**

7 References **38**

A Distributions Used **40**

B Properties of $f(\Theta \mid a + \alpha)$ **43**

List of Figures

3.1 A simple example of a belief network. 7

4.1 Random Network Case Study: $t = 20$, $m = 50$ 17
4.2 Random Network Case Study: $t = 20$, $m = 100$ 17
4.3 Random Network Case Study: $t = 20$, $m = 150$ 18
4.4 Random Network Case Study: $t = 20$, $m = 200$ 18
4.5 Alarm Network Case Study: $m = 50$ 20
4.6 Alarm Network Case Study: $m = 150$ 20
4.7 Insurance Network Case Study: $m = 50$ 21
4.8 Insurance Network Case Study: $m = 150$ 21

5.1 Examples of Error Bars 31
5.2 Testing the validity of the normal approximation. 33

List of Tables

4.1 Results for Random Networks 19
4.2 Results for Alarm Network 20
4.3 Results for Insurance Network 21

5.1 Results for the Diamond Graph 32
5.2 Results for the Alarm Network 33
5.3 Results for Random Graphs 34

Chapter 1

Introduction

Belief networks (also known as Bayesian networks, graphical models) are commonly used to model joint probability distributions in expert systems. A graph is used to represent all direct dependencies between the variables. This structure both organizes reasoning and reduces the dimensionality of the parameter space to the point where it is feasible to estimate parameters from sample data. When network parameters have been learned from a random sample, those parameters are themselves random variables, adding an additional level of uncertainty, which may be represented by a posterior distribution over parameter values (Bayesian perspective) or by the sampling distribution of the unknown true parameters (Frequentist perspective).

The uncertainty in parameters is an issue when learning network structure. One can typically obtain an improved fit to the data by increasing the complexity of the structure, and thus the dimensionality of the parameter space; however, this increased representational power comes at a cost, namely the increase in parameter variance. Finding the appropriate balance between complexity and goodness of fit is a matter of handling this *bias-variance trade-off*. The trade-off may be operationalized as a scoring function on network structures known as a *model selection criterion*. We argue that there is no one criterion which is the *right* one — rather, the choice of a model selection criterion should reflect available prior knowledge and the goals of the learner. Prior knowledge and utilities are often implicit, however, and consequently difficult to formalize. Therefore, it is important to understand the issues involved in model selection, and how different criteria behave in different contexts. To this end, we compare a number of proposed criteria for learning belief network structure. While we cannot endorse any particular criterion as superior to the others, we demonstrate the differences between the criteria that are relevant for making a choice between them.

The uncertainty in parameters is also an issue when carrying out inference based on a network model. An inference is a function of (some of) the parameters in a network, and, being a function of random variables, is itself a random variable. For a confidence/credibility level of $(1-\delta)$, error-bars around an estimate specify a range of values "containing" $(1-\delta)$ of the probability mass under the distribution of that variable. Providing such error-bars for belief network inferences is a way to quantify the uncertainty that they inherit from the parameters. Error-bars are less informative than giving the complete distribution around an inference, but they are typically easier to compute and reason with. Such error-bars can be extremely useful, for example, when taking actions based on network inferences, or when choosing whether or not to obtain additional evidence at a cost. In this dissertation we describe a method for efficiently computing approximate error-bars for inferences, and we give empirical results to demonstrate the effectiveness of our method.

Overview

Chapter 2 covers related work and presents our notation. Chapter 3 is an introduction to belief networks, which covers their representation, basic properties, and the posterior and sampling distributions of network parameters that we refer to throughout the paper. Chapter 4 describes our work on model selection: 4.1 presents the theoretical background for the model selection problem; 4.2 describes the setup of our empirical study of model selection criteria; 4.3 presents the results; and 4.4 concludes with a discussion of the results and unresolved issues. Chapter 5 describes our work on error-bars for belief network inferences: 5.1 reviews the notions of confidence regions and credible regions; 5.2 gives our derivation of approximate error-bars for belief network inference; 5.3 describes our empirical study and describes our results. Chapter 6 concludes the paper, summarizing our results and giving directions for future work. Appendix A lists the properties of the distributions that we use, for ease of reference, and Appendix B gives the derivations of some properties of belief networks that are used throughout the paper.

Chapter 2

Preliminaries

2.1 Related Work

2.1.1 Belief Networks

Early work on induction in AI tended to focus on symbolic manipulation and logical (truth-functional) representations. Pearl (1988) was instrumental in directing attention to probabilistic methods, and, in particular, to belief networks. Basic research on the properties of belief networks followed; Cooper and Herskovits (1992) is a source of considerable insights. As is typical of research in any field, once basic results were established there was a flowering of additional research exploring refinements, applications and specializations. Topics that have received attention include: sensitivity analysis, hidden variables, incomplete data, parameter estimation, structure learning, continuous variables, approximate inference, efficient inference, and applications for image analysis, medical diagnosis and decision making, to name a few. Belief networks are subsumed by the more general notion of *influence diagrams*, which may include test nodes and decision nodes as well as evidence variables.

2.1.2 Model Selection Criteria

There is a considerable literature on learning belief networks, and in particular, on learning their structure; see Heckerman (1995) for a detailed overview of the subject. Note that many researchers, including Lam and Bacchus (1994) Suzuki (1996), and Friedman and Goldszmidt (1996) explicitly use the MDL criterion (or something close to it) to evaluate candidate networks. We explore the behaviour of this MDL criterion, among others. Friedman and Yakhini (1996) carry out an analysis of the sample requirements for various complexity penalty approaches to belief net learning. While that work also addresses suitability of various selection criteria, its analysis is theoretical and based on asymptotic behaviour, and it only considers complexity penalization; by contrast we are empirically investigating small sample behaviour over a different class of criteria, including Bayesian, bootstrap and cross-validation criteria.

Linhart and Zucchini (1986) provide an overview of the general problem of model selection, covering AIC and cross-validation, but not MDL. Rissanen (1989) gives a detailed development of the *Minimum Description Length Principle*, which is the information-theoretic view of induction that the MDL criterion is based on. Schwarz (1978) gives an alternative derivation of the MDL criterion (therein referred to as BIC, the *Bayesian Information Criterion*) as a large-sample approximation of the Bayesian posterior criterion. Bozdogan (1987) gives the derivation of AIC and a discussion of its use. Kearns et al (1997) describe experiments similar to our own. They make a similar comparison between an MDL criterion and a cross-validation criterion, and report similar behaviour in a different context: learning a function $f : [a, b] \to \{0, 1\}$ from noisy examples, under zero-one loss.

2.1.3 Error-Bars

Our results provide ways to compute the variance of a belief network inference as a function of the variance of the individual parameters (under the sampling or posterior distribution, depending on perspective). This is done using the ***delta-method*** (Bishop et al 1995): approximately propagating the variance of each parameter, based on first partial derivatives. Kleiter (1996) performs a similar computation. His analysis is more general, in that he considers incomplete data; however, he does not provide a closed form for the derivative, nor does he provide an efficient way to compute this information. Moreover, our empirical studies provide additional evidence that the approximations inherent in this approach are appropriate.

A related topic is ***sensitivity analysis***, which involves propagating "ranges" of parameter values to produce ranges in the responses (see Che et al 1993, Laskey 1995, Castillo et al 1997 and Darwiche 2000). These papers deal with the case where the user can simply specify the "range" of a parameter value; by contrast our work considers the source of these variances based on a data sample. In addition, our analysis involves propagating the variance of all the parameters, whereas most other analyses consider only propagating a single range. Moreover, except for the recent work by Darwiche (2000), none of these other projects provides a closed-form expression for the partial derivative nor does any provide an efficient way to compute that information. Also, some of those other papers focus on properties of this derivative; for example, when it is 0 for a particular parameter. Note that this information follows immediately and implicitly from our expression for the derivative and our means of computing it. Finally, our analysis holds for arbitrary structures; by constrast some other results (eg. Che et al 1993) deal only with singly connected networks (trees).

2.2 Notation

The notation used to express statements about probability distributions and random variables is frequently a source of confusion. Here we will introduce a notation that deviates slightly from existing conventions for reasons of clarity.

We need to represent random variables, probability distributions over random variables and the parameters of those distributions. Our formulae may be complicated by the fact that variables are often vector valued (discrete or continuous), and that a parameter may in certain cases be a random variable as well. Therefore, we will drop the distinction between parameters, variables, and random variables, and simply denote all variables with names that begin with upper case Roman or Greek letters. The names of symbolic constants will be in lower case Roman letters or Greek letters. A variable name in bold font will denote the set of values it may take on. So X is a random variable that takes a value $x \in \mathbf{X}$. In addition, we will occasionally use script letters, such as $\mathcal{X}$, to denote an arbitrary assignment (such as $X = x$).

We will denote functions as their "signatures" — by an application to variables. Where variables are replaced by constants, this denotes a new function of lesser arity — a function of zero arity is a constant. Function parameters may be divided into two groups by the "conditioning bar", which means that if all the parameters to the left of the bar are fixed while those to the right are free to vary, then the function is a probability mass or density function. Obviously, where variables are discrete, distributions are probability mass functions, otherwise they are probability density functions. To illustrate this with an example, suppose that we have a function $P(X, Y \mid A, B)$. $P(X, Y \mid a, b)$ denotes a distribution, and $P(x, y \mid a, b)$ denotes a probability (or probability density).

We introduce the following notation for probability statements. $\Pr[g(X) = x;\ P(X)]$ denotes the probability that $g(X) = x$ under the distribution $P(X)$; in general, any statement is allowed on the left. We will denote the expectation and variance of $g(X)$ with respect to a distribution $P(X)$ as $\mathrm{E}[g(X);\ P(X)]$ and $\mathrm{Var}[g(X);\ P(X)]$ respectively, with

SD[$g(X)$; $P(X)$] being the standard deviation. The frequency that X is assigned x in a random sample d will be denoted Fr[$X = x$; d]. The conditioning bar will be used to denote conditional sentences appearing on the left in any of these expressions. When the reference distribution is clear from context, we will omit it.

Chapter 3

Belief Networks

3.1 Representation

A belief network is a representation for a joint distribution over a set of random variables $X = \langle X_1 \ldots X_n \rangle$. It consists of a dependency graph and a set of conditional probability functions. The dependency graph is a directed acyclic graph, whose vertices are the random variables. This *network structure* represents the assumption that the joint distribution can be written as:

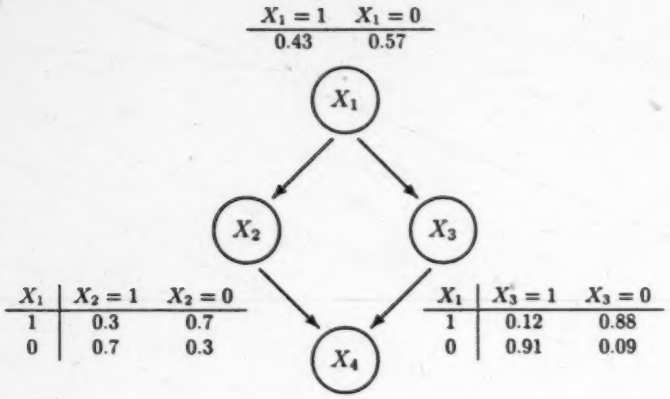
$$P(X_1 \ldots X_n) = \prod_{i=1}^{n} P_i(X_i \mid W_i)$$

where each $P_i(X_i \mid W_i)$ is the local, conditional distribution of X_i, given its parents in the dependency graph, W_i. Formally, the network structure implies that X_i is independent of all non-descendents, given its parents. The resulting set of conditional independencies allows for a compact representation when the network is sparse.

In the most general case, the network variables may be continuous or discrete, and the conditional probability functions may be represented in a variety of ways. In this paper we are restricting our attention to discrete, finite-valued variables, and extensionally represented conditional probability functions, whose parameters are an exhaustive set of conditional probabilities that may be directly estimated from sample frequencies. In addition, we will assume that our sample data is composed of complete tuples (all network variables assigned a value).

The simplest representation for the conditional probability functions is as *CP-tables* (see Figure 3.1). For a variable X_i, X_i's CP-table is represented by a number of *rows*, one for each possible assignment to W_i. Each row specifies a conditional distribution of X_i as a multivariate Bernoulli distribution (see Appendix A). The parameters of these individual distributions, taken together, are the parameters of the network. In some cases, greater economy of representation can be attained by marginalizing over rows of a CP-table, representing the CP-table as a tree structure (see Friedman & Goldszmidt, 1996). The analysis of parameter variance that follows holds for this latter approach as well.

We have introduced the notation $P(X)$ to denote the distribution given by a belief network model. Because we will often need to refer to families of distributions over X we need to introduce further notation. We will use $P(X \mid \Theta)$ to denote the parameterized class of models given by a belief network structure, where Θ ranges over parameter vectors (CP-table entries). We will also have occasion to use $P(X \mid \Theta, H)$ to denote all possible belief networks on X, where H ranges over network structures — note that Θ is constrained by H.



X_1	$X_2 = 1$	$X_2 = 0$
1	0.3	0.7
0	0.7	0.3

X_1	$X_3 = 1$	$X_3 = 0$
1	0.12	0.88
0	0.91	0.09

X_2	X_3	$X_4 = 1$	$X_4 = 0$
1	1	1.00	0.00
1	0	0.81	0.19
0	1	0.50	0.50
0	0	0.22	0.78

Figure 3.1: A simple example of a belief network.

3.2 Parameter Estimation

A dependency graph defines a parameterized class of distributions: $P(X \mid \Theta)$, where $\Theta = \langle \Theta_1 \ldots \Theta_k \rangle$ is the vector of parameters (probabilities in the CP-tables). Suppose we are given a sample $d = \langle d_1 \ldots d_m \rangle$ of complete assignments to X, drawn i.i.d. from the true distribution. The likelihood function is given by:

$$L(d \mid m, \Theta) = \prod_{i=1}^{m} P(d_i \mid \Theta)$$

Next we define a sufficient statistic for Θ. Let $a = \langle a_1 \ldots a_k \rangle$ denote the conditional sample frequencies for the parameters $\langle \Theta_1 \ldots \Theta_k \rangle$. That is, when $\langle \Theta_r \ldots \Theta_s \rangle$ are the parameters of some CP-table row giving the probability distribution over X_j for an assignment $W_j = w$ to its parents, and X_j takes values from the set $\{x_r \ldots x_s\}$, then $\langle a_r \ldots a_s \rangle$ are the frequencies $a_i = \mathrm{Fr}[X_j = x_i, W_j = w; d]$ for $i = r$ to s. We will later use b_i to denote $(a_r + \cdots + a_s) - a_i$, for $i = r$ to s; that is, $b_i = \mathrm{Fr}[X_j \neq x_i, W_j = w; d]$. Given the i.i.d. assumption and a fixed network structure, a is a sufficient statistic for Θ — that is, if two different samples have the same value of a then those samples have the same likelihoods under any model within the class of models defined by the structure.

Maximum likelihood estimation is the classic Frequentist approach to fitting parameters. The maximum likelihood estimate for Θ is the vector of conditional sample proportions $\hat{\theta} = \langle \hat{\theta}_1 \ldots \hat{\theta}_k \rangle$, where for $i = 1$ to k:

$$\hat{\theta}_i = \frac{a_i}{a_i + b_i}$$

Note that this is not always defined — because the parameters represent conditional probabilities it is possible that no data may be observed that matches the conditioning event,

and thus $(a_i + b_i)$ may sometimes be 0.

The maximum a posteriori (MAP) estimate is the Bayesian counterpart to the maximum likelihood estimate, and the two are equivalent when the prior is uniform. However, an ideal Bayesian approach to prediction involves averaging over all possible parameter values, weighted according to the posterior distribution. This is known as the predictive distribution. For a natural family of belief network priors, the predictive distribution is easily computed; it is the mean of the posterior. Below we state two important theorems that make the Bayesian analysis of belief networks tractable. Their proofs are given in Appendix B.

Theorem 1 *Conjugate Property (Cooper & Herskovits 1992)* Assume a fixed network structure, and suppose that the prior distribution over the parameter space is given by a product of Dirichlet distributions (see Appendix A), one for each CP-table row. Recall that each row is a multivariate Bernoulli distribution which gives the distribution of a variable for an assignment to its parents. Thus, if $\langle \Theta_r \ldots \Theta_s \rangle$ are the parameters of some row, then they are jointly Dirichlet distributed with parameters $\langle \alpha_r \ldots \alpha_s \rangle$. Denote the prior as $f(\Theta \mid \alpha)$, where $\alpha = \langle \alpha_1 \ldots \alpha_k \rangle$ is the concatenation of the prior parameters for all the individual Dirichlet distributions.

Given the above assumptions, the posterior distribution is $f(\Theta \mid \alpha + a)$, where a is the sufficient statistic defined previously, and $(\alpha + a)$ is simple vector addition. Thus, f defines a *conjugate* family of distributions over Θ.

Theorem 2 *Model Averaging (Cooper & Herskovits 1992)* Assume a fixed network structure, and a posterior distribution of the form described above. Then the predictive distribution:

$$Q(X) = \int_{\Theta} P(X \mid \Theta) f(\Theta \mid \alpha + a)\, d\Theta$$

is the average model, $\bar{\theta} = \langle \bar{\theta}_1 \ldots \bar{\theta}_k \rangle$, where, for $i = 1$ to k:

$$\bar{\theta}_i = \frac{\alpha + a}{(\alpha + a + \beta + b)}$$

Here β has the analogous relationship to α that b has to a — namely, when $\langle \Theta_r \ldots \Theta_s \rangle$ are jointly distributed, for $i = r$ to s: $\beta_i = (\alpha_r + \cdots + \alpha_s) - \alpha_i$.

Chapter 4

Comparing Model Selection Criteria

4.1 Theoretical Background

Most learning problems can be cast in the following form. First, a *type* of model (such as Markov model, belief network, neural network, decision tree, etc.) is chosen, based on (often tacit) domain knowledge. Next, within this model type a ***structure*** is chosen. This structure defines a parametric class of models. Last, the parameters are estimated, or tuned/fit, based on a sample. The problem of choosing the structure, or parametric class of models, is called the ***model selection*** problem. Model selection can be viewed as an optimization problem where there are two separate issues: (1) how to search the space of structures, and (2) what function (model selection criterion) to optimize for. Here we explore the second issue: the choice of a model selection criterion.

4.1.1 The Bias-Variance Trade-off

A standard approach to parameter estimation is to maximize the likelihood of the data. Applying this principle to model selection, however, tends to result in the phenomenon of *overfitting*. A more complex structure defines a larger class of models, and thus one cannot decrease the maximum of the likelihood function by going to a more complex structure. A higher dimensional parameter space allows for a tighter fit to the data, but increases the variance in the parameter estimates. There are two sources of true error: (1) the error due to *bias*, which can be defined as the true error of the optimal (under the true distribution) parameters for a structure, and (2) the variance component of error, which can be defined as the expected additional error due to the estimation of the parameters from a random sample. Bias-error results from choosing a structure that is insufficient to represent the true model; variance-error results from parameter estimates being suboptimal. The trade-off between these two sources of error is called the ***bias-variance trade-off***. When a model-selection procedure chooses a structure that is larger than the optimal one, we say it is *overfitting*. When a model-selection procedure chooses a structure that is smaller than optimal, we say it is *underfitting*.

In this section we are concerned with two kinds of bias: (1) the bias in an error estimate, and (2) the bias of a structure, or model class. Here we define these two notions.

Bias of an Estimator: The difference between the expected value of an estimate (under the true sampling distribution) and the true value. We say an estimator is biased when its expectation differs from the true value.

Bias of a Model Class: The minimum true error of any model in the class.

Suppose we have an unbiased parameter estimation process. Then the expectation of the parameter estimate is the lowest true error model for the structure. For belief networks, maximum-likelihood parameter estimation is unbiased (regardless of the true parameters). In this case, the bias-error of the learned model is entirely due to the bias of the structure. From a Bayesian perspective, however, we may prefer a biased estimator for the parameters, such as the mean of the posterior. In this case, the bias-error of the resulting model will be due to both the bias of the structure and the bias of parameter estimation. We will discuss this issue in more detail later.

It is widely believed that one ought to handle the bias-variance trade-off through model selection, attempting to optimize for true error by selecting a structure of just the right complexity. Since bias and variance depend on the true distribution we are trying to learn, however, it is not clear how we can acheive this. In fact, overfitting avoidance is generally not justifiable unless one has a prior expectation of simplicity, or one is interested in something else besides expected error, such as a pragmatic desire for low variance or simple structures. In practice, avoiding overfitting seems to be desirable, perhaps because the choice of model type reflects a prior belief that a parsimonious representation exists for this type. See Wolpert (1996a, 1996b) for more on this topic, which we will return to later in this section.

4.1.2 Error Functions

A standard measure of training error is the negative log-likelihood:

$$\mathrm{DL}(d, \hat{\theta}, h) = -\log L(d \mid m, \hat{\theta}, h)$$

where $d = \langle d_1 \ldots d_m \rangle$ is a sample of size m, h is a model structure, $\hat{\theta}$ is the parameter vector for that model, and $L(d \mid m, \hat{\theta}, h)$ is the likelihood function.[1] We denote it as DL because it is the description length of the data given an optimal code based on $P(X \mid \hat{\theta}, h)$. When d is an i.i.d. sequence of values $d_1 \ldots d_m$ then:

$$\mathrm{DL}(d, \hat{\theta}, h) = \sum_{i=1}^{m} -\log P(d_i \mid \hat{\theta}, h)$$

KL-divergence (due to Kullback & Leibler, 1951; see also Cover & Thomas, 1991) is a widely used measure of true error for distribution learning. If $P(X \mid \theta, g)$ is the "true" model, and $P(X \mid \hat{\theta}, h)$ is a hypothesized model, then the KL-divergence of $P(X \mid \hat{\theta}, h)$ from $P(X \mid \theta, g)$ is given by:

$$\mathrm{Err}[P(X \mid \theta, g);\ P(X \mid \hat{\theta}, h)] = \sum_{X} P(X \mid \theta, g) \log \frac{P(X \mid \theta, g)}{P(X \mid \hat{\theta}, h)}$$

This is the expected cost of encoding instances from $P(X \mid \theta, g)$ using a code based on $P(X \mid \hat{\theta}, h)$. Note that it can also be written as:

$$\mathrm{Err}[P(X \mid \theta, g);\ P(X \mid \hat{\theta}, h)] = \mathrm{E}[\mathrm{DL}(X, \hat{\theta}, h)] - \mathrm{E}[\mathrm{DL}(X, \theta, g)]$$

where the expectations are taken under $P(X \mid \theta, g)$. The second term is the *entropy* of $P(X \mid \theta, g)$. As it does not depend on $P(X \mid \hat{\theta}, h)$, the first term alone is sufficient to compare models. We can form an estimate of this first term:

$$\mathrm{Fit}(h, \hat{\theta}, d) = \frac{1}{m}\mathrm{DL}(d, \hat{\theta}, h)$$

The problem, however, is that if we use d to estimate $\hat{\theta}$ then $\hat{\theta}$ depends on d and so $\mathrm{Fit}(h, \hat{\theta}, d)$ is a biased estimator, underestimating $\mathrm{E}[\mathrm{DL}(X, \hat{\theta}, h)]$. As h grows more complex, the dimensionality of $\hat{\theta}$ increases and the more it can be tuned to d, increasing the bias in $\mathrm{Fit}(h, \hat{\theta}, d)$.

[1] We leave the estimation of $\hat{\theta}$ as a black box for the time being — we return to this at the end of the section.

Note that there are two different kinds of bias involved here: as models become more complex, they become less representationally biased (they can represent a larger class of distributions), but their parameters have higher variance under sampling, and so $\text{Fit}(h, \hat{\theta}, d)$ becomes *more* biased as an estimator of $\text{E}[\text{DL}(X, \hat{\theta}, h)]$. Model selection criteria attempt to correct for this bias or avoid it altogether.

4.1.3 Complexity Penalty Criteria

One way to (attempt to) correct for the bias in $\text{Fit}(h, \hat{\theta}, d)$ is to add a complexity penalty to the function. The major difficulty here is determining what an appropriate penalty is. One cannot determine a priori how much bias there is in $\text{Fit}(h, \hat{\theta}, d)$, as this depends on $P(X \mid \theta, g)$ as well as $P(X \mid \hat{\theta}, h)$. For each structure, there exists a continuum of possible biases in $\text{Fit}(h, \hat{\theta}, d)$, from zero, when the true model has zero entropy and zero variance, on up to some maximum value, when the true model has maximum entropy. Therefore, one cannot justify a complexity penalty simply on the grounds that it is a bias correction; one must appeal to other considerations.

There are several well-known penalty functions, each motivated by different theoretical considerations. The Minimum Description Length (MDL) criterion is based on an information-theoretic view of induction as data compression; see Rissanen (1989) for a detailed development. It is equivalent to the *Bayesian Information Criterion* (BIC), which was introduced originally by Schwarz (1978) and given a Bayesian interpretation. The information-theoretic interpretation of the MDL criterion is as the length of an encoding of the sample as a two part code. The model defines a code for the sample, and one encodes the sample by first encoding the model, and then encoding the data using the optimal code given by the model.[2] If the model captures significant features of the data, this encoding will be considerably smaller than the original encoding of the sample. On the other hand, if the model represents too much about the sample, the encoding size will increase (Linhart & Zucchini, 1986). This trade-off is similar to the bias-variance trade-off.

The MDL criterion we will use is given by:

$$\text{MDL}(h, d) = \text{Fit}(h, \hat{\theta}, d) + \frac{\text{Dim}(h) \log m}{2m}$$

where $\text{Dim}(h)$ is the the number of *free* parameters of h. Recall that m is the sample size. This version differs from the standard form in that we have normalized everything by $1/m$ so we can compare it across sample sizes and with other criteria. Some low order terms (all positive) have been dropped as well (from the MDL criterion given by Rissanen; the BIC is exactly what we have above). It will be seen that this omission has no negative impact on the criterion.

Akaike's Information Criterion (AIC) comes from a different theoretical perspective. It is an explicit attempt to correct the overfitting bias. See Bozdogan (1987) for the derivation of the criterion. The complexity penalty is considerably smaller than MDL's. Our version of AIC is given by:

$$\text{AIC}(h, d) = \text{Fit}(h, \hat{\theta}, d) + \frac{\text{Dim}(h) \log e}{m}$$

where e is the base of the natural logarithm, and $\log e$ converts from nats to bits.

4.1.4 Validation Criteria

An alternative approach is to use part of the data to estimate the parameters and the rest of the data to estimate the error of the resulting model. There are drawbacks to this approach, however. The training sample used to evaluate a structure is smaller than the sample that will ultimately be used to train the chosen structure, and this results in a preference for

[2] Every probability distribution has an associated optimal code (Cover & Thomas 1991).

simpler structures. (Once a structure has been chosen, all available data are used to estimate the parameters of the final model.) Also, there will be increased variance in the criterion, as opposed to the naive $\mathrm{Fit}(h, \hat{\theta}, d)$, because of the smaller test-sample size. So this *train-test* validation trades the bias of the naive criterion for another bias and increased variance.

Cross-validation (Stone 1974) is a more sophisticated approach which seeks to alleviate these concerns. In cross-validation every datum in the sample is used both for training and testing. The sample is partitioned into r subsamples, each of which is used once for testing and $(r-1)$ times for training. Each subsample is tested on the model that results from training on the remaining $(r-1)$ subsamples. This is called *r-fold* cross validation. The most extreme case is m-fold (*leave-one-out*) cross-validation. We will use the simplest version, 2-fold cross-validation, which is given by:

$$\mathrm{XV}(h, d) = \frac{1}{m}\left[\mathrm{DL}(d_1 \ldots d_r, \hat{\theta}(d_{r+1} \ldots d_m), h) + \mathrm{DL}(d_{r+1} \ldots d_m, \hat{\theta}(d_1 \ldots d_r), h)\right]$$

where $r = \lfloor m/2 \rfloor$ and $\hat{\theta}(\cdot)$ denotes the parameter estimated from the data given in the parentheses.

Cross-validation has its problems as well. First of all, the meaning of the criterion is not explicit — it is an average of many individual estimates which are not independent if $r > 2$. Leave-one-out cross-validation is not *asymptotically minimal*: in the limit of sample size it may not prefer the simplest unbiased hypothesis (Stone 1977). Worst of all, it may be very expensive to compute, and time is at a premium when searching a large space of structures.

4.1.5 Bootstrap Criteria

Bootstrap methods are similar to validation methods, but instead of withholding data from the original sample, a training or test sample is generated by *resampling* (Efron 1982). The idea is that, by adding noise through resampling, the variance of the original sampling process is simulated. One problem with the non-parametric bootstrap is that the empirical variance of the bootstrapped sample tends to be lower than that of the original sample — if the bootstrap is recursively applied over and over, with asymptotic probability 1 it will result in a homogenous sample. Another problem is that the empirical distribution in the sample will differ from the true distribution, and thus will have a different variance under sampling.[3] The formula below shows the bootstrap criterion we will be using, where the training sample has been bootstrapped and the original sample is used for testing.

$$\mathrm{Boot}(h, d) = \mathrm{Fit}(h, \hat{\theta}(d'), d)$$

Here d' is a sample of size m, generated from d by resampling with replacement.

4.1.6 A Bayesian Approach?

Overfitting results from the naive maximum-likelihood approach because the likelihood of the data is based on a single (maximum-likelihood) model in the model space for each structure. As this space increases, the mode of the likelihood function cannot decrease, and tends to increase. The Bayesian approach to prediction, by contrast, involves averaging over all models according to the posterior distribution over models. For a given structure, one can compute the marginal likelihood of the data by integrating over the parameter space. The integral under the likelihood function (coupled to the prior distribution over models), rather than its mode, may thus be used as a way to evaluate structures.[4] This mitigates the effects of increasing complexity: as complexity increases the likelihood function becomes more peaked, increasing its mode but potentially decreasing the integral underneath after a

[3] It might be possible to correct for some of these problems by measuring and correcting for the difference in variance between the two samples.

[4] Prior distributions are implicit where not mentioned.

point. The bias-variance trade-off is handled implicitly — it is reflected in the shape of the likelihood function.

This, however, is a Frequentist justification for a putatively Bayesian procedure. The notion of overfitting is not really defined in the Bayesian paradigm: upon seeing the data, you simply compute the posterior distribution over models and use this to carry out inference. A possible Bayesian approach is to parameterize the model space by first defining a hyper-parameter that ranges over structures; then the marginalized likelihood is the data-dependent part of the posterior distribution over structures. Ideally, prediction would involve marginalizing over all structures according to this posterior, but for pragmatic reasons one could approximate this by choosing the structure with the maximum posterior probability, given the data. (See Heckerman 1996.)

The use of a hyper-parameter ranging over structures, however, must be viewed simply as a device to define the prior over models. The model class for the most complex structure contains every possible model, and so "structure" is a superfluous syntactic notion. A uniform prior over structures, coupled with a uniform prior over the parameters for each structure, results in a very *non-uniform* prior over the space of models, favouring those models which appear in more structures (exactly those representable in simpler structures). Perhaps this is desirable, being a reflection of the intuition behind choosing a particular model type and representational framework. But it is open to the criticism of being ad hoc, as well as the classic objection to Bayesian methods as being subjective — because many incompatible uniform priors may be proposed as representing identical states of ignorance (see Howson 1997).

One could carry out a Bayesian analysis of model selection as a decision problem, where the objective is to minimize expected loss. Where KL-divergence is the loss function, and $f(\Theta, G \mid d)$ is the posterior over models, the problem reduces to computing:

$$\begin{aligned}
&\operatorname{argmin}_{h,\vartheta} \int_{\mathbf{G}} \int_{\boldsymbol{\Theta}} \sum_{X} \left[\begin{array}{r} P(X \mid \Theta, G) \log P(X \mid \Theta, G) \\ -P(X \mid \Theta, G) \log P(X \mid \vartheta, h) \end{array} \right] \cdot f(\Theta, G \mid d)\, \mathrm{d}\Theta\, \mathrm{d}G \\
&= \operatorname{argmin}_{h,\vartheta} \int_{\mathbf{G}} \int_{\boldsymbol{\Theta}} \left[\begin{array}{r} \sum_X P(X \mid \Theta, G) \log P(X \mid \Theta, G) \\ -\sum_X P(X \mid \Theta, G) \log P(X \mid \vartheta, h) \end{array} \right] \cdot f(\Theta, G \mid d)\, \mathrm{d}\Theta\, \mathrm{d}G \\
&= \operatorname{argmin}_{h,\vartheta} \int_{\mathbf{G}} \int_{\boldsymbol{\Theta}} -\sum_X P(X \mid \Theta, G) \log P(X \mid \vartheta, h) \cdot f(\Theta, G \mid d)\, \mathrm{d}\Theta\, \mathrm{d}G \\
&= \operatorname{argmin}_{h,\vartheta} -\sum_X \int_{\mathbf{G}} \int_{\boldsymbol{\Theta}} P(X \mid \Theta, G) f(\Theta, G \mid d)\, \mathrm{d}\Theta\, \mathrm{d}G \cdot \log P(X \mid \vartheta, h)
\end{aligned}$$

Based on a fundamental result of information theory due to Shannon (see Cover & Thomas 1991), a minimum is attained at the entropy of the predictive distribution when:

$$P(X \mid \vartheta, h) = \int_{\mathbf{G}} \int_{\boldsymbol{\Theta}} P(X \mid \Theta, G) f(\Theta, G \mid d)\, \mathrm{d}\Theta\, \mathrm{d}G$$

Thus, the Bayes-optimal strategy for minimizing KL-divergence is to choose the predictive distribution, taken over all models.

This is a sort of "no free lunch" theorem (Wolpert 1996a) for model selection, as it follows that any attempt to avoid overfitting reflects a prior bias toward simplicity. For example, if one's prior belief is reflected by a uniform prior over all possible models, one ought to use the *most complex* structure, with a uniform prior over its parameter space, and then take the average model from the posterior that results from Bayesian updating, which is the predictive distribution for this prior. Of course, one might be interested in other properties of a learning algorithm besides expected loss (like minimax properties, for example).

It is interesting to note that the marginalized likelihood has an interpretation as a validation criterion. For belief networks the marginalized (predictive) likelihood (for a fixed structure) is given by (see Appendix B):

$$\int_{\boldsymbol{\Theta}} L(d \mid m, \Theta) f(\Theta \mid \alpha)\, \mathrm{d}\Theta = \prod_{i=1}^{m} \int_{\boldsymbol{\Theta}} P(d_i \mid \Theta) f(\Theta \mid \alpha + a(d_1 \ldots d_{i-1}))\, \mathrm{d}\Theta$$

where $a(\cdot)$ is the sufficient statistic for Θ, given the data in the parentheses. Note that the data, though independently distributed for a fixed model, are not independent when integrating over all models.

This *prequential* criterion (Dawid & Vovk, 1999) involves iteratively computing the probability of each datum given those already seen by marginalizing over the parameters using the current distribution over parameter space, then updating that distribution with the current datum. Each datum is used once for testing and once for training, but the testing precedes the training, so each datum's error score is based on parameters it has not been used to estimate, as in validation criteria. For belief networks it may be efficiently computed, because using the average model under the posterior distribution is equivalent to integration over the parameter space (again, see Appendix B for details). We use the following form as a model selection criterion:

$$\text{Preq}(h, d) = \frac{1}{m} \sum_{i=1}^{m} -\log P(d_i \mid \bar{\theta}(d_1 \ldots d_{i-1}), h)$$

where $\bar{\theta}(d_1 \ldots d_{i-1})$ is the mean parameter vector after updating with $d_1 \ldots d_{i-1}$.

4.1.7 KL-Divergence and Parameter Estimation

Maximum-likelihood estimation is not the ideal approach to minimizing KL-divergence. Assigning a probability of 0 to any event with non-zero true probability makes the KL-divergence infinite, whereas maximum-likelihood estimation will always assign probability mass to observed data rather than unobserved data, if the parameterization allows it. For example, if we were to flip a coin twice and observe heads both times, we could maximize likelihood by assigning a probability of 1 to heads and a probability of 0 to tails. If tails had even a very small true probability of occurring, however, our model would have infinite KL-divergence. As the model complexity grows, and parameters are estimated based on fewer data, it becomes increasingly likely that the KL-divergence for a maximum-likelihood estimate is infinite.

It is therefore unrealistic to use maximum-likelihood estimates when comparing criteria under the KL-divergence loss function. Therefore, in our experiments we will use the average model (predictive distribution) for each structure, under uniform priors: $\bar{\theta}_i = (a_i+1)/(b_i+r)$ for all i, when $\bar{\theta}_i$ is the CP-table entry $P_j(X_j = x \mid W_j = w)$ and X_j ranges over r values. This replaces $\hat{\theta}$ in all the criteria.

One consequence of using this parameter estimation method is that the asymptotic arguments used to justify the AIC and MDL criteria no longer strictly hold; the AIC is designed to correct the bias in maximum-likelihood estimates, while the MDL criterion is based on optimizing the precision in the parameters, assuming maximum-likelihood estimation. Another consequence is that $\text{Fit}(h, \hat{\theta}, d)$ is not necessarily monotonically decreasing as model complexity increases — using the average model instead of the maximum likelihood model has a smoothing effect which avoids some overfitting automatically. In certain cases, as we will see, this is sufficient to remove the bias in $\text{Fit}(h, \hat{\theta}, d)$.

4.2 Empirical Study

We carried out a series of experiments where we chose a true model, drew a sample from it, and evaluated the criteria across a set of hypothesis network structures. We also computed the true error of each structure with the parameters estimated from the sample. This allowed us to compare the behaviour of the criteria across a range of situations, by comparing their evaluations to the true error function (in particular, comparing the minima).

4.2.1 Criteria

We compared the following criteria described in the previous section:

- The "Bayesian" prequential criterion (Preq).
- 2-fold cross-validation (XV).
- A bootstrap criterion (Boot).
- Akaike's information criterion (AIC).
- The Minimum Description Length criterion (MDL).

Each criterion takes a hypothesis structure and a sample of m data as input, and returns a real number being the estimated error per datum. Each uses m training iterations and m testing iterations, where a training iteration is Bayesian updating with a single datum, and a testing iteration is computing the negative log-likelihood of a single datum. Thus, all required the same amount of computational resources, and each assumed that the average model (predictive distribution) would be used for each structure.

4.2.2 True Models

We considered the following domains:

- Random networks: We generated networks on 10 binary variables with 10, 20, and 30 links. In each case, we generated the parameters from a uniform distribution over the parameter space.
- The Alarm network (Bienlich et al 1989): This is a benchmark network commonly used in empirical studies on belief networks, constructed by medical experts for monitoring patients in intensive care units. It has 37 variables each with 2-4 possible values, 46 links (very sparse), and 509 parameters.
- The Insurance network (Binder et al 1997): This is another widely used benchmark network. It represents car insurance risks. It has 27 variables, 52 links and over 1400 parameters.[5]

4.2.3 Hypotheses

For each true model, we generated sets of hypotheses as follows. Each set of hypotheses was a sequence of network structures that included the true structure. Starting from the simplest network, with no links, each hypothesis in the sequence was constructed from the previous by adding a link. Eventually the true structure appeared, followed by progressively more complex networks with "redundant" links. (We constructed these sequences from the true structure by randomly deleting and adding links.) By construction, the model class defined by each hypothesis properly contained the model classes of all preceding hypotheses. Thus, the bias with respect to the true model was decreasing up to the point where the true structure appeared, and after that all error could be attributed to variance. We did this so that we could observe the behaviour of the criteria as bias decreased and variance increased.

4.2.4 Experimental Design

Each experiment consisted of the following steps:

1. Fix a true model.
2. Generate a sequence of candidate network structures.

[5]The Alarm and Insurance networks are available at the UCI machine learning repository.

3. Generate a sample from the true distribution.

4. Evaluate the criteria on that sample, across all structures.

5. Compute the sample error, Fit$(h, \tilde{\theta}, d)$, and the true error Err$[P(X \mid \theta, g);\ P(X \mid \tilde{\theta}, h)]$ for each structure.

We looked at individual experiments as well as gathering summary statistics from large numbers of experiments — the case studies allowed us to identify patterns of behaviour; the comprehensive studies allowed us to quantify some aspects of those patterns. Since we were only interested in the issue of the criteria used for model selection, not the search strategy, we based our comprehensive results on selecting the structure that minimized each criterion, then comparing the true error of those preferred structures.

4.3 Results

Figure 4.1 shows an experiment on a randomly generated true model, for a sample of size 50. Here the true distribution was represented by a 20-edge dependency graph on 10 binary variables, where each conditional distribution was randomly generated from a uniform Beta distribution (see Appendix A). The x-axis shows the complexity of the hypothesis networks in terms of number of links, while the y-axis shows the error in terms of bits. The entropy of the true distribution has been subtracted from all y-values to scale them down and to allow for comparisons between different true distributions.[6] In each graph all criteria are plotted, as are the true error (Err) and the sample error (Fit).

From left to right across the graph, the hypothesis complexity is increasing, and the true error decreases until it reaches a minimum near the true structure ($t = 20$), where it begins to climb back up. We can observe the bias-variance trade-off in action: bias decreases until the 20-link network, while variance is increasing continuously. The sample error (Fit), here shown as the dashed line, continues to fall past the true structure, showing that a naive maximum-likelihood approach to choosing a structure would lead to overfitting here. We observed the criteria across the range of hypotheses, for different sample sizes ($m = 50, 100, 150, 200$). Figures 4.2–4.4 show the same experiment as the sample size is increased. Ideally, we would like a criterion to have the same shape, or at least the same minimum, as the true error function. By observing how the criteria differ from this unrealizable ideal, in different learning contexts, we get an idea of how they handle the bias-variance trade-off.

Both the MDL and AIC criteria tend to underfit at first, and then converge on the true network as more data are provided. MDL, in particular, has a strong bias for simplicity that requires a lot of data to counteract. In addition, a small number of additional data can radically change the evaluation of a structure — notice the difference in the AIC scores from Figure 4.1 to Figure 4.2. The bootstrap criterion, by contrast, tends to overfit, a tendency which does not diminish as sample size increases. It appears to reflect sample error as much as or more than it reflects true error. The prequential criterion and 2-fold cross-validation give very close evaluations, across the range of hypotheses and sample sizes. They also match best with the shape of the true error function, having minima and maxima at identical or nearby structures. The prequential criterion appears to be slightly less sensitive to underlying fluctuations in the true error.

In addition to randomly generated distributions, we considered "natural" distributions defined by benchmark networks. Here we give results on two widely used benchmarks: the Alarm and Insurance networks. The former has 37 variables and 46 links; the latter 27 variables and 52 links. Figures 4.5 and 4.6 show the pattern observed on the Alarm network, while Figures 4.7 and 4.8 show the same results for the Insurance network. First of all,

[6] Note that this makes some of the scores negative, as the empirical entropy tends to underestimate true entropy, but this does not affect our comparisons.

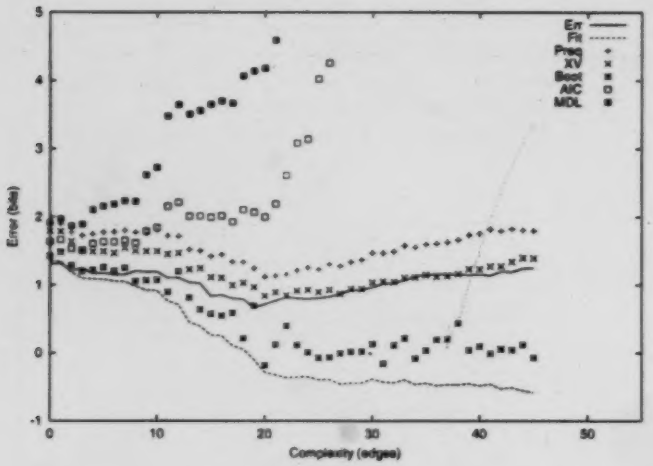


Figure 4.1: Random Network Case Study: $t = 20$, $m = 50$

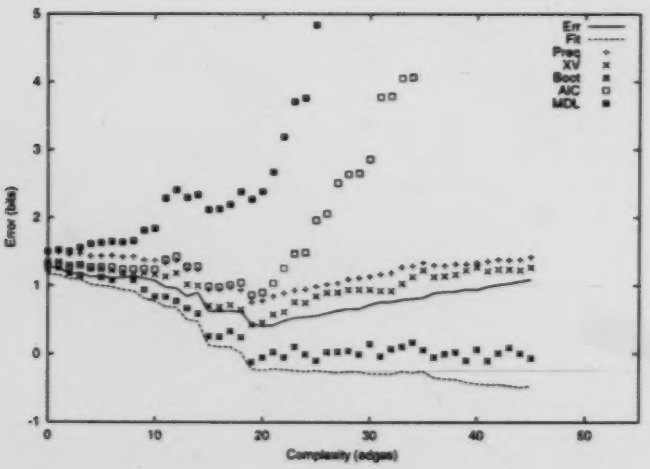


Figure 4.2: Random Network Case Study: $t = 20$, $m = 100$

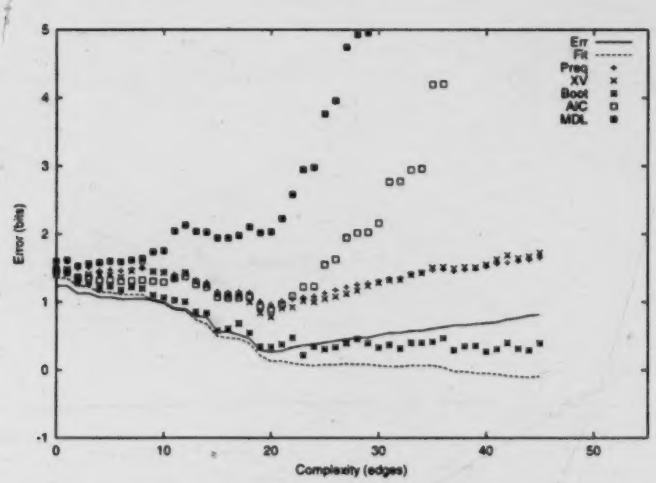


Figure 4.3: Random Network Case Study: $t = 20$, $m = 150$

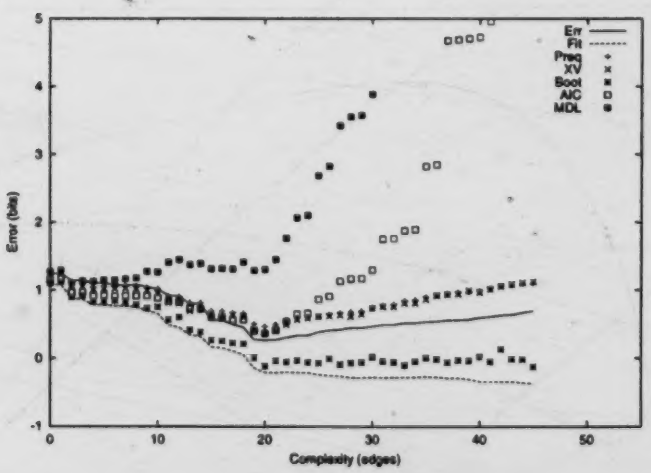


Figure 4.4: Random Network Case Study: $t = 20$, $m = 200$

Table 4.1: Results for Random Networks

t	m	Fit	Preq	XV	Boot	AIC	MDL
10	50	0.673424	0.045774	0.085914	0.526975	0.087324	0.253013
10	100	0.547661	0.005703	0.025173	0.454687	0.017614	0.130488
10	150	0.487156	0.007373	0.016239	0.363291	0.008755	0.047132
10	200	0.453187	0.003962	0.007292	0.311022	0.003405	0.020010
20	50	0.524959	0.058641	0.104019	0.390290	0.479635	0.765795
20	100	0.451487	0.020641	0.037211	0.352972	0.157028	0.643342
20	150	0.432322	0.015944	0.020506	0.304437	0.093406	0.486259
20	200	0.370868	0.003800	0.017478	0.305357	0.061886	0.317950
30	50	0.255416	0.047017	0.101118	0.177463	0.765639	0.981904
30	100	0.259497	0.016463	0.021493	0.173163	0.598727	0.969809
30	150	0.244873	0.008740	0.027909	0.158132	0.455095	0.889597
30	200	0.223711	0.008547	0.016779	0.163472	0.357849	0.866408

note that the sample error (Fit) is not monotonically decreasing as hypothesis complexity increases. If the maximum likelihood estimate $\hat{\theta}$ were used, instead of $\bar{\theta}$, then Fit would necessarily be monotonic, because by construction each successive hypothesis structure defines a class of models which contains the model class of all predecessors. The effect observed here is due to the smoothing effect of choosing the *average* a posteriori model within each class, instead of the *maximum* a posteriori (or maximum-likelihood) model. This is interesting because many papers have been published describing results of structure-learning experiments on the Alarm and Insurance networks, where various methods were used to avoid overfitting, (see Liu et al, 1998, for example) when in fact it is virtually impossible to overfit on these networks.[7] We believe that the difference observed here between the random networks and these benchmark networks is due to the fact that the benchmark networks are locally low entropy — the conditional probability distributions for each node tend to be highly skewed. On these networks all the data-dependent criteria worked fairly well, including the bootstrap, because its determination by the sample error was not a deficit. The complexity-penalty criteria underfit; again, the MDL criterion was much slower to converge than AIC.

In our comprehensive studies we attempted to measure the difference in error that would result from using each criterion to pick a structure. We carried out experiments of the form described above, but instead of plotting the criteria across all hypotheses, we used each criterion to select a hypothesis by taking its argmin across the hypotheses under consideration. We then compared the true error of this preferred hypothesis (for each criterion) with the minimum true error obtained by any hypothesis (the ideal criterion, of course, being the true error). The additional true error of the preferred hypothesis was our dependent variable. On random networks we carried out experiments for several combinations of truth complexity (t) and sample size (m). For each assignment to t and m we carried out 30 experiments, generating a new true model and hypothesis sequence each time. We summarize these experiments by giving the average additional true error for each criterion. Table 4.1 displays our results for random networks; Tables 4.2 and 4.3 present our results on the Alarm and Insurance networks; where we carried out identical experiments to those described on random networks, except of course that the true model was fixed. Again, each cell represents 30 experiments.

[7] This also demonstrates the perils of over-reliance on benchmark problems.

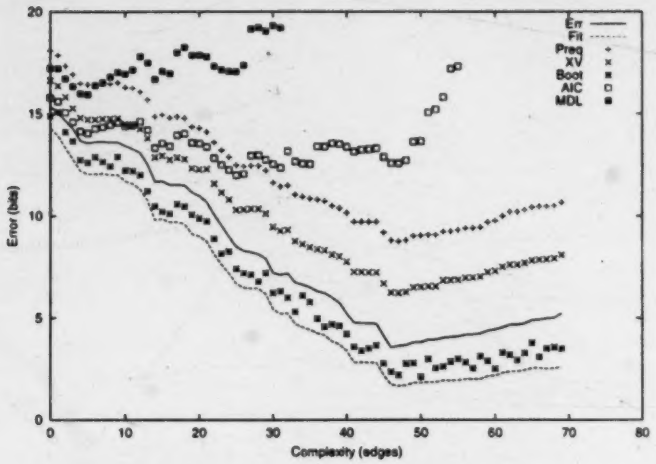


Figure 4.5: Alarm Network Case Study: $m = 50$

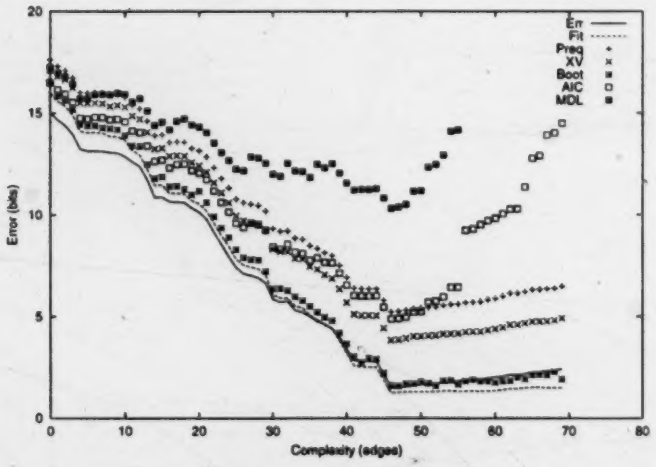


Figure 4.6: Alarm Network Case Study: $m = 150$

Table 4.2: Results for Alarm Network

m	Fit	Preq	XV	Boot	AIC	MDL
50	0.002055	0.002055	0.002055	0.130730	3.386938	9.996832
100	0.007444	0.000000	0.000473	0.094789	0.000000	4.657260
150	0.008488	0.000608	0.002463	0.045147	0.000608	0.000000
200	0.003116	0.000000	0.000582	0.033719	0.000000	0.000000

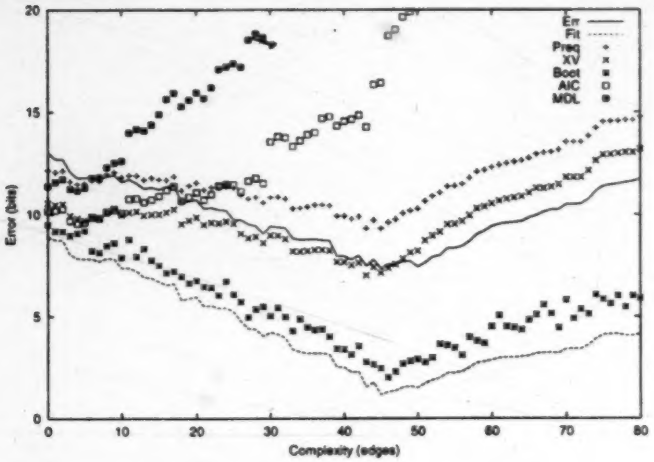


Figure 4.7: Insurance Network Case Study: $m = 50$

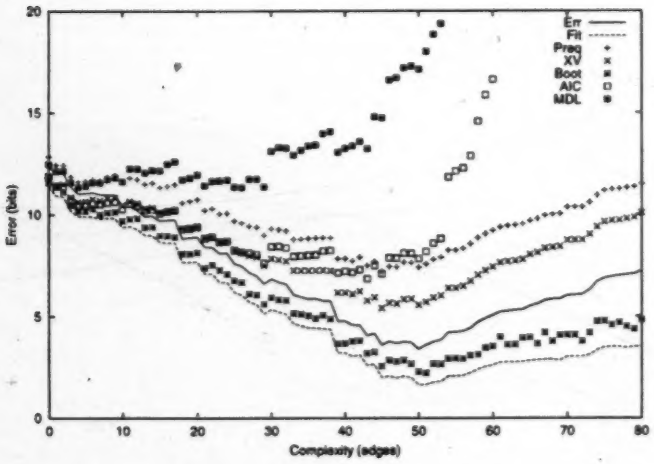


Figure 4.8: Insurance Network Case Study: $m = 150$

Table 4.3: Results for Insurance Network

m	Fit	Preq	XV	Boot	AIC	MDL
50	0.190568	0.171794	0.126868	0.273680	4.692191	4.768936
100	0.010237	0.025124	0.024360	0.086121	1.243281	6.565789
150	0.000000	0.167360	0.129376	0.057607	0.669900	3.597372
200	0.010371	0.243592	0.147450	0.071585	0.512019	3.464004

4.4 Discussion

Model selection involves dealing with fundamental aspects of inductive inference. The no free lunch theorems show that there is no justification for avoiding overfitting that is independent of prior beliefs and/or pragmatic considerations. In other words, we avoid overfitting either because we have prior beliefs that the truth tends to be simple, or because we prefer simpler models for pragmatic reasons — they are easier to remember, explain and reason with. Ideally, we could pose model selection as a Bayesian decision problem — in the first case our bias for simplicity would be reflected in the prior; in the second case our bias for simplicity would be reflected in the loss function. However, it is often the case that we have no clear idea of how to represent our intuitions in the prior or loss function. For this reason, it is of interest to consider properties of various proposed model selection criteria, to see how well or how poorly they capture our intuitions.

From our observations, we conclude that the MDL and AIC criteria display a strong bias for simplicity that may require considerable data to outweigh. This is particularly true for MDL. The amount of data required is much more than the amount of data required by other criteria to get a good estimate of the true error of the structure by other criteria. Also, a small amount of additional data can radically affect the evaluation of a structure, and it is impossible to predict in advance how much data is required to evaluate a given structure (as this depends on the true distribution).

In defense of the MDL criterion, however, it was not derived specifically to avoid overfitting (although that is a benefit frequently claimed for it), but simply to implement the MDL principle, which takes an information-theoretic view of induction orthogonal to the Bayesian and Frequentist paradigms.[8] Furthermore, it was designed with asymptotic consistency and minimality in mind, not small sample behaviour. Nevertheless, one should be aware when applying the MDL principle in this context that one may end up seriously underfitting the data.

The prequential criterion and 2-fold cross-validation displayed similar behaviour to one another. Both converged quite rapidly in shape to the form of the true error function. This makes them good functions to optimize for if one's goal is minimizing expected error and one's prior expectations of simplicity are modest. The bootstrap criterion, by contrast, proved to be a better estimator of sample error than of true error. Perhaps our formulation was naive and a better one could be obtained (for minimal additional computation?), however, we would not recommend the use of a bootstrap criterion based on these results.

Aside from the behaviour of particular criteria, we observed some interesting phenomena relating to the general problem of model selection in belief networks. Very little data seems to be required to get a good evaluation of a network, even if the true distribution is reasonably complex; for example, a sample size of less than 50 was sufficient to get a good evaluation of structures for the Alarm network, which has 37 variables and 46 dependencies between them. This is surprising, given that recent work in the area of learning belief network structure (see Liu et al, 1998, for example) uses as many as 10,000 data to learn this network. As search spaces for model selection are very large, one could probably do much better by evaluating more networks on fewer data. Also, we noted that the practice of taking the average model (which is Bayes-optimal within a fixed model class) handles overfitting to some extent. In fact, although the Alarm network has been used in several studies of structure-learning, using the sample error of the average model to choose a structure results in *no overfitting* for this learning problem. This is explained by the smoothing effect of the prior, which increases as the model complexity increases.

We cannot conclude from this research that any one criterion is better than another, unless we specify exactly what loss function and prior beliefs we have. We can suggest caution in applying complexity-penalty criteria such as MDL and AIC (particularly MDL), as they may lead to underfitting the data — they appear to be risky criteria. In some situations they are dominated by the complexity penalty, and thus almost independent of

[8] We can criticize it as the BIC, however.

the data. This is probably not desirable in most learning contexts. The bootstrap criterion, on the other hand, seems to lead to overfitting, as it is governed more by the sample error than by the true error. The prequential criterion and 2-fold cross-validation seem to be relatively risk-free and assumption-light criteria suitable for a wide range of problems.

It might be possible to rehabilitate the complexity penalty criteria or the bootstrap criterion by a more careful formulation. Recall that we were not using maximum-likelihood parameter estimates because in most cases this results in infinite KL-divergence for more complex hypotheses. We used the predicitive distribution instead, which is the Bayes-optimal model within the model class for each structure. The MDL and AIC criteria were designed for maximum-likelihood estimates (although this is a subtlety that is frequently ignored). On the other hand, these criteria cannot be viewed as correcting bias anyway, for it is impossible to predict, a priori, the bias in sample error as an estimate of true error. (This can only be established asymptotically, which, as we have seen, is a rather meaningless assertion, especially when we can derive two different criteria this way.) Our bootstrap criterion also might be improved, by correcting somehow for the expected reduction in the empirical variance of the bootstrapped sample. These would merely be attempts to make these criteria more like the cross-validation and prequential criteria, however, so the value of this exercise is not clear.

4.4.1 Future Work

There are many avenues for future research. It would be interesting to consider other loss functions, other representations for belief networks, other model types such as Markov models or decision trees, and additional criteria. The experimental framework we have developed could be used to explore these and other issues. As well, we would like to carry out a more thorough theoretical analysis of the model selection problem in general. There are other interesting properties of learning algorithms besides expected error. We would like to compare these criteria in terms of asymptotic consistency, minimality, and convergence rates. Minimax properties are another interesting basis for a comparison: if an adversary could choose the true model, knowing your model selection strategy, how could you minimize the expected error? It would be worthwhile to complement the perspective taken here by these alternative theoretical perspectives.

Chapter 5

Error-Bars for Inference

5.1 Error-Bars

Two standard approaches to providing error-bars for a point estimate are Bayesian *credible regions* and Frequentist *confidence regions*. Suppose $u(\theta) \in \mathbf{U}$ is a deterministic function of an unknown parameter $\theta \in \Theta$.

Credible Region: Let $f(\Theta)$ be the posterior distribution of Θ. A $(1-\delta)$ credible region for $u(\theta)$ is a region ω in $\mathbf{U}$ for which $\Pr[u(\Theta) \in \omega;\ f(\Theta)] = (1-\delta)$.

Confidence Region: Let $L(D \mid m, \theta)$ be the likelihood function for a sample D, given θ. A $(1-\delta)$ confidence region $u(\theta)$ is a region $\omega(D)$ in $\mathbf{U}$ such that $\Pr[u(\theta) \in \omega(D);\ L(D \mid m, \theta)] = (1-\delta)$. Here $\omega(D)$ is a statistic (function of the data) for which this equality holds *before d is fixed by sampling*. After observing d, however, $\Pr[u(\theta) \in \omega(d)]$ is independent of $L(d \mid m, \theta)$, and depends only on the true value of θ. Note that, because θ is unknown, the equality $\Pr[u(\theta) \in \omega(D);\ L(D \mid m, \theta)] = (1-\delta)$ has to hold irrespective of the value of θ.

The logic of confidence regions is somewhat more subtle than that of credible regions, and they are sometimes wrongly interpreted as being statements about posterior probability. Confidence regions may also be much more difficult to derive than credible regions. Here we present a Bayesian approach to error-bars, deriving *approximate credible intervals* for belief network inferences. Assuming that the network structure is fixed, we show how to propagate the uncertainty about the parameter values through the inference process to derive a measure of the uncertainty of an inference.

There are two major difficulties in deriving a credible interval: (1) deriving the form of the posterior distribution, and (2) integrating under it. Even though we have a conjugate prior for Θ and computing the posterior for Θ is trivial, it is non-trivial to derive the posterior of an inference based on Θ. Instead, we use an approximation based on the *delta-method:* propagating the variance of the parameter vector through the vector of first partial derivatives of the inference with respect to the parameters (see Bishop et al 1995). We give a closed form expression for these derivatives, and describe an algorithm to compute them efficiently. This gives us an approximation to the variance of an inference. Also, as the approximate error is a linear combination of parameters, we use a normal approximation to its distribution, which enables us to compute the interval from the variance and mean of the inference, using tables for the Standard Normal distribution. Finally, we give empirical evidence that our method produces accurate error-bars in a variety of situations.

5.2 Derivation

A belief network may be used to answer arbitrary queries about the distribution it models: any conditionalization or marginalization of the joint distribution may be the object of a

query. A belief network query thus takes the form $\Pr[\mathcal{Q} \mid \mathcal{E}]$, where $\mathcal{Q}$ and $\mathcal{E}$ are assignments to subsets of the network variables, called the *query variables* and *evidence variables* respectively. If we fix $\mathcal{Q}$ and $\mathcal{E}$ but allow Θ to vary, then we can consider an inference to be a function $q(\Theta)$, which has a posterior distribution under $f(\Theta \mid \alpha + a)$, the posterior for Θ.

For an arbitrary query response $q(\bar{\theta})$, the actual error of this value is $q(\theta) - q(\bar{\theta})$, when θ is the true value of the parameter. (Recall that $\bar{\theta}$ is the mean of the posterior distribution of Θ, also the predictive model.) Likewise, the actual error in $\bar{\theta}$ is $\theta - \bar{\theta}$. Since θ is unknown, we consider the random variable $\mathrm{Err} = q(\Theta) - q(\bar{\theta})$, and how it depends on $\bar{\theta}$ and the distribution of Θ.

Taking the Taylor series expansion of $q(\Theta)$ at $\bar{\theta}$ gives us:

$$q(\Theta) = q(\bar{\theta}) + [\Theta - \bar{\theta}]^T \cdot q'(\bar{\theta}) + [\Theta - \bar{\theta}]^T \cdot q''(\bar{\theta}) \cdot [\Theta - \bar{\theta}]/2 + \cdots$$

Thus, we can derive an expression for $q(\Theta) - q(\bar{\theta})$:

$$\mathrm{Err} = [\Theta - \bar{\theta}]^T \cdot q'(\bar{\theta}) + [\Theta - \bar{\theta}]^T \cdot q''(\bar{\theta}) \cdot [\Theta - \bar{\theta}]\,/2 + \cdots$$

to which a linear approximation is given by:

$$\widetilde{\mathrm{Err}} = [\Theta - \bar{\theta}]^T \cdot q'(\bar{\theta})$$

where $q'(\bar{\theta})$ is the vector of first partial derivatives evaluated at $\bar{\theta}$; $q''(\bar{\theta})$ is the matrix of second partial derivatives, and the superscript ^{T} denotes transposition. This expands to the sum:

$$\widetilde{\mathrm{Err}} = (\Theta_1 - \bar{\theta}_1) \cdot q_1'(\bar{\theta}) + \cdots + (\Theta_k - \bar{\theta}_k) \cdot q_k'(\bar{\theta})$$

where each $q_i'(\bar{\theta})$ is the ith partial derivative evaluated at $\bar{\theta}$.

Assuming that the parameters are all independent (we know this is not necessarily true), and the variance of each parameter Θ_i is σ_i^2 then each $(\Theta_i - \bar{\theta}_i)$ is a random variable with mean 0 and variance σ_i^2. It follows that $\widetilde{\mathrm{Err}}$ is a random variable, and being a linear combination of bounded random variables, it may be approximated by a Normal distribution with mean 0 and variance given by:

$$\mathrm{Var}[\widetilde{\mathrm{Err}};\ f(\Theta \mid \alpha + a)] = \sum_{i=1}^{k} \sigma_i^2 \cdot q_i'(\bar{\theta})^2$$

Thus we have derived an expression for an *approximate* $(1 - \delta)$ error bar, $\pm\varepsilon$, such that:

$$\Pr[|\ \widetilde{\mathrm{Err}}\ | > \varepsilon;\ f(\Theta \mid \alpha + a)] < \delta,$$

by choosing ε so that $\Phi(-z) = \delta/2$, where Φ is the standard normal distribution function, and $z = \varepsilon/\mathrm{SD}[\widetilde{\mathrm{Err}};\ f(\Theta \mid \alpha + a)]$.

5.2.1 Handling Parameter Dependencies

If the parameters are not all independent then (a) we cannot take true partial derivatives, and (b) the Normality of $\widetilde{\mathrm{Err}}$ is thrown into question. We can deal with the linear effects of such statistical dependencies, however, by multiplying the derivatives through the variance-covariance matrix of the parameters, namely:

$$\mathrm{Var}[\widetilde{\mathrm{Err}};\ f(\Theta \mid \alpha + a)] = q'(\bar{\theta})^T \cdot \sigma \cdot q'(\bar{\theta})$$

where σ is the variance-covariance matrix of $f(\Theta \mid \alpha + a)$ and $q'(\bar{\theta})$ is the vector of first partial derivatives, evaluated at $\bar{\theta}$ (see Bishop 1995, pp. 292–297). For the case where $f(\Theta \mid \alpha + a)$ factors into independent Dirichlet distributions over the parameters of each CP-table row, we need only consider the variance-covariance matrix for each row, as all other covariances are 0. The form of the variance-covariance matrix for a Dirichlet distribution is given in Appendix A.

5.2.2 Partial Derivatives of Belief Net Queries

Consider the belief net query $\Pr[\mathcal{Q} \mid \mathcal{E}] = q(\tilde{\theta})$, where $\mathcal{Q}$ and $\mathcal{E}$ are assignments to arbitrary subsets of the network variables. Here we give a closed form expression for the partial derivative of $q(\tilde{\theta})$, with respect to an arbitrary parameter $\tilde{\theta}_i = \Pr[\mathcal{B} \mid \mathcal{A}]$.

Lemma 1 The partial derivative of $\Pr[\mathcal{C}]$ with respect to a network parameter $\Pr[\mathcal{B} \mid \mathcal{A}]$ is $\Pr[\mathcal{C} \mid \mathcal{B}, \mathcal{A}] \cdot \Pr[\mathcal{A}]$.

Proof:

$$\Pr[\mathcal{C}] = \sum_{a \in \mathbf{A}} \sum_{b \in \mathbf{B}} \Pr[\mathcal{C} \mid B = b, A = a] \cdot \Pr[B = b \mid A = a] \cdot \Pr[A = a]$$

$\Pr[\mathcal{C} \mid \mathcal{B}, \mathcal{A}] \cdot \Pr[\mathcal{B} \mid \mathcal{A}] \cdot \Pr[\mathcal{A}]$ is one of the terms of this summation, and thus the partial derivative of $\Pr[\mathcal{C}]$ with respect to $\Pr[\mathcal{B} \mid \mathcal{A}]$ is $\Pr[\mathcal{C} \mid \mathcal{B}, \mathcal{A}] \cdot \Pr[\mathcal{A}]$, as claimed.

Theorem 3 The partial derivative of a query $\Pr[\mathcal{Q} \mid \mathcal{E}]$ with respect to a network parameter $\Pr[\mathcal{B} \mid \mathcal{A}]$ is given by:

$$\frac{\partial \Pr[\mathcal{Q} \mid \mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]} = (\Pr[\mathcal{A}]/\Pr[\mathcal{E}]) \cdot (\Pr[\mathcal{Q}, \mathcal{E} \mid \mathcal{B}, \mathcal{A}] - \Pr[\mathcal{Q} \mid \mathcal{E}] \cdot \Pr[\mathcal{E} \mid \mathcal{B}, \mathcal{A}])$$

Proof: $\Pr[\mathcal{Q} \mid \mathcal{E}] = \Pr[\mathcal{Q}, \mathcal{E}]/\Pr[\mathcal{E}]$, thus,

$$\frac{\partial \Pr[\mathcal{Q} \mid \mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]} = \frac{\Pr[\mathcal{E}] \cdot \frac{\partial \Pr[\mathcal{Q}, \mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]} - \Pr[\mathcal{Q}, \mathcal{E}] \cdot \frac{\partial \Pr[\mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]}}{\Pr[\mathcal{E}]^2}$$

It follows from Lemma 1 that:

$$\frac{\partial \Pr[\mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]} = \Pr[\mathcal{E} \mid \mathcal{B}, \mathcal{A}] \cdot \Pr[\mathcal{A}]$$

and:

$$\frac{\partial \Pr[\mathcal{Q}, \mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]} = \Pr[\mathcal{Q}, \mathcal{E} \mid \mathcal{B}, \mathcal{A}] \cdot \Pr[\mathcal{A}]$$

thus:

$$\frac{\partial \Pr[\mathcal{Q} \mid \mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]} = \frac{\Pr[\mathcal{E}] \cdot \Pr[\mathcal{Q}, \mathcal{E} \mid \mathcal{B}, \mathcal{A}] \cdot \Pr[\mathcal{A}] - \Pr[\mathcal{Q}, \mathcal{E}] \cdot \Pr[\mathcal{E} \mid \mathcal{B}, \mathcal{A}] \cdot \Pr[\mathcal{A}]}{\Pr[\mathcal{E}]^2}$$

and so:

$$\frac{\partial \Pr[\mathcal{Q} \mid \mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]} = (\Pr[\mathcal{A}]/\Pr[\mathcal{E}]) \cdot (\Pr[\mathcal{Q}, \mathcal{E} \mid \mathcal{B}, \mathcal{A}] - \Pr[\mathcal{Q} \mid \mathcal{E}] \cdot \Pr[\mathcal{E} \mid \mathcal{B}, \mathcal{A}])$$

as claimed.

5.2.3 Efficient Computation of Partial Derivatives

Computing the derivative directly from the formula given above would result in an excruciatingly slow algorithm — useless for all practical purposes. Below we give an algorithm for *efficiently* computing the partial derivatives of a given query $\Pr[\mathcal{Q} \mid \mathcal{E}]$ with respect to all network parameters. (Here "efficient" means not appreciably more time-consuming than the inference itself — belief net inference is known to be NP-hard.) The method described below takes advantage of any heuristics available to speed up inference. First we describe an approach to belief network inference known as *bucket elimination*, then we describe an algorithm for computing derivatives based on this inference process, and last we discuss other inference methods.

Bucket Elimination

Bucket elimination, described in (Dechter 1996), is an elegant framework for belief net inference, and also subsumes a number of inference procedures in other domains. It is a dynamic programming algorithm which iteratively eliminates variables by marginalization. Below we describe bucket elimination.

First we need to introduce some notation. We will use $\lambda S.u$ to denote a *table*, or function. S is the *scheme* of the table, which is a set of named variables. The table is a mapping from all possible assignments to S to the reals: $\lambda S.u : \mathbf{S} \rightarrow \Re$. Next we need to define two operations on tables, elim and join.

Join is an operation that combines several tables into a new table:

$$\lambda T.v = \text{join}\{\lambda S_1.u_1 \ldots \lambda S_r.u_r\}$$

when $T = \bigcup_{i=1}^{r} S_i$ (recall that the variables are named), and

$$(\lambda T.v \quad t) = \prod_{i=1}^{r} (\lambda S_i.u_i \quad S_i : t)$$

where each $S_i : t$ denotes the assignment of the variables of S_i to the corresponding values in t. In other words, $\lambda T.v$ maps from each assignment to T to the product of the $\lambda S_i.u_i$ on the subassignments to their schemes. Joining tables creates a new and possibly much larger table denoting the product of the original tables, where same-named variables are unified. This operation is similar to the relational join used in database theory (see Korth et al 1998).

Elimination is an operation on a table that reduces its scheme by marginalizing over a set of variables.

$$\lambda T.v = \text{elim}_R[\lambda S.u]$$

when $T \subseteq S$, $R \subseteq S$, $T = S - R$ and:

$$(\lambda T.v \quad t) = \sum_{r \in \mathbf{R}} (\lambda S.u \quad S : t : r)$$

where $S : t : r$ denotes the assignment of the variables of S to their corresponding values in t and r. In other words, eliminating the variables R from a table $\lambda S.u$ creates a new table mapping from a reduced scheme, which maps from an assignment t to $\lambda S.u$ evaluated at t, summed over the domains of the remaining variables. If $R = \emptyset$ then $\text{elim}_R[\lambda S.u] = \lambda S.u$.

Now consider the function:

$$y = \text{elim}_{(\bigcup_{i=1}^{n} S_i)} [\text{join}\{\lambda S_1.p_1 \ldots \lambda S_n.p_n\}]$$

where $X = \bigcup_{i=1}^{n} S_i$ are the variables of a belief network, and the $\lambda S_i.p_i$ are the CP-tables. Recall that each CP-table maps from a subset of the network variables. Here the join represents the full joint distribution, as a mapping from all possible assignments to X; and since all variables have been eliminated, $y : \quad \rightarrow 1$. As such, it is not an interesting function, but, if we restrict $\mathbf{X}$ by fixing X on some dimensions (say, restricting $\mathbf{X}_3$ to $\{0\}$ and $\mathbf{X}_5$ to $\{1\}$) then computing y amounts to computing any marginalization of the full joint (such as $\Pr[X_3 = 0, X_5 = 1]$). Still not very interesting, though, because we are computing these marginalizations by constructing a table for the full joint and then marginalizing over it — not a very efficient process. Bucket elimination is a means for computing such a function efficiently, by eliminating variables one at a time. We describe the algorithm below.

To compute y for a given partial assignment to X, first fix X on those dimensions that are assigned. Then create a sequence of "buckets", $b_0 \ldots b_n$, one for each dimension of X, and one *terminal* bucket, b_0. Next, order these buckets according to a heuristic (left as a black box), where b_0 always comes last in the ordering. Then place each table $\lambda S_i.p_i$ into

the *first* bucket in the ordering, b_j, such that $X_j \in S_i$, or b_0 if $S_i = \emptyset$ (for belief network inference this never happens). Note that the subscript on b_j refers to the dimension j, not b_j's position in the ordering.

Now iterate over the buckets in order, processing them as follows. Suppose b_j contains the tables $\{\lambda M_1.u_1 \ldots \lambda M_r.u_r\}$. Recall that in this collection, each M_i contains X_j. Create a new table:

$$\lambda T_j.v_j = \text{elim}_{\{X_j\}}\,[\text{join}\{\lambda M_1.u_1 \ldots M_r.u_r\}]$$

and place this table into the next bucket in the ordering, b_h, such that $X_h \in T_j$.

When this process has run to completion, the result is obtained by processing the terminal bucket, b_0, which contains constants (functions whose scheme is null).

This procedure reduces the complexity of computing y by taking advantage of the independencies between the tables. It requires time proportional to the *induced tree width* of the dependency graph between the variables, given the bucket ordering (hence the need for a good ordering heuristic — see Dechter 1996 for more details). There are some additional modifications which can improve efficiency (rather than marginalize over some functions, we can ignore them as they are guaranteed to sum to 1), but these details are not relevant here.

To apply this algorithm to an arbitrary belief network inference, $\Pr[\mathcal{Q} \mid \mathcal{E}]$, we compute $\Pr[\mathcal{Q}, \mathcal{E}]$ and $\Pr[\mathcal{E}]$ using bucket elimination, and then divide to get the answer. Next we describe an extension to bucket elimination that computes the derivatives of the initial functions.

Computing the Derivatives

We want to compute the partial derivative of y with respect to each value in a CP-table. Suppose $\lambda S_i.p_i$ is a CP-table, and s is an assignment to S_i — then $(\lambda S_i.p_i \quad s)$ is a network parameter. For each CP-table, we can represent these derivatives as another function, $\lambda S_i.y'_{p_i}$, where:

$$(\lambda S_i.y'_{p_i} \quad s) = \frac{\partial y}{\partial(\lambda S_i.p_i \quad s)}$$

for all $s \in \mathbf{S}$. We show how to construct $\lambda S_i.y'_{p_i}$ by giving a proof of the following proposition.

Theorem 4 Every function $\lambda S.u$ that is placed in a bucket has a counterpart function $\lambda S.y'_u$ such that:

$$y = \text{elim}_S\,[\text{join}\{\lambda S.u, \lambda S.y'_u\}]$$

Proof: We prove by induction on the buckets.

Basis: Prove for all tables placed in b_0. Recall that $y = \text{elim}_\emptyset[\text{join}\{\forall u \text{ in } b_0\}]$. Clearly, for all u in b_0:

$$y'_u = \text{elim}_\emptyset\,[\text{join}\{\forall v \text{ in } b_0, v \neq u\}]$$

Inductive Step: Consider an arbitrary bucket b_j. Let $\lambda T_j.v_j$ be the result of processing b_j. Suppose by way of induction that we have the function $\lambda T_j.y'_{v_j}$ (note that $\lambda T_j.v_j$ was placed in a bucket appearing *after* b_j in the ordering). Let the contents of b_j at the time of processing be $\{\lambda M_1.u_1 \ldots \lambda M_r.u_r\}$. We will show how to construct $\lambda M_h.y'_{u_h}$ for an arbitrary $\lambda M_h.u_h$. Recall that:

$$\lambda T_j.v_j = \text{elim}_{\{X_j\}}\,[\text{join}\{\lambda M_1.u_1 \ldots \lambda M_r.u_r\}]$$

and also that:

$$y = \text{elim}_{T_j}\left[\text{join}\{\lambda T_j.v_j, \lambda T_j.y'_{v_j}\}\right]$$

by assumption. It follows that:

$$y = \text{elim}_{T_j}\left[\text{join}\{\lambda T_j.y'_{v_j}, \text{elim}_{\{X_j\}}\,[\text{join}\{\lambda M_1.u_1 \ldots \lambda M_r.u_r\}]\}\right]$$

The following thus gives y as a value (rather than a function):

$$y = \sum_{t \in \mathbf{T}_j} (\lambda T_j.y'_{v_j} \quad t) \sum_{x \in \mathbf{X}_j} \prod_{i=1}^{r} (\lambda M_i.u_i \quad M_i : t : x)$$

$$y = \sum_{(t,x) \in \mathbf{T}_j \times \mathbf{X}_j} (\lambda T_j.y'_{v_j} \quad T_j : t) \prod_{i=1}^{r} (\lambda M_i.u_i \quad M_i : t : x)$$

Recall that $M_h \subseteq T_j \cup \{X_j\}$. Thus:

$$y = \sum_{m \in \mathbf{M}_h} \sum_{s \in \mathbf{S}} (\lambda T_j.y'_{v_j} \quad T_j : m : s) \prod_{i=1}^{r} (\lambda M_i.u_i \quad M_i : m : s)$$

where S is the scheme $(T_j \cup \{X_j\}) - M_h$. Consequently,

$$y = \sum_{m \in \mathbf{M}_h} (\lambda M_h.u_h \quad m) \sum_{s \in \mathbf{S}} (\lambda T_j.y'_{v_j} \quad T_j : m : s) \prod_{i=1, i \neq h}^{r} (\lambda M_i.u_i \quad M_i : m : s)$$

Therefore:

$$\lambda M_h.y'_{u_h} = \mathrm{elim}_S \left[\mathrm{join} \left(\{\lambda T_j.y'_{v_j}\} \cup \{M_i.u_i : i \in \{1 \ldots r\}, i \neq h\} \right) \right]$$

which proves our claim.

The constructive nature of our proof immediately gives us an algorithm: iterate over the buckets in the reverse order of their original processing, constructing $\lambda M.y'_u$ for every $\lambda M.u$ placed in each bucket, and thus constructing the first derivative of each of the original tables. And because we iterate over the buckets in the reverse order of processing, we have already computed $\lambda T_j.y'_{v_j}$ when we arrive at b_j, and thus a single pass over the buckets suffices to compute all the derivatives.

Now, assuming that we respond to the query $\Pr[\mathcal{Q} \mid \mathcal{E}]$ by computing $\Pr[\mathcal{Q}, \mathcal{E}]$ and $\Pr[\mathcal{E}]$ using bucket elimination, we then compute $\frac{\partial \Pr[\mathcal{Q}, \mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]}$ and $\frac{\partial \Pr[\mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]}$ for each belief net parameter, $\Pr[\mathcal{B} \mid \mathcal{A}]$, and obtain $\frac{\partial \Pr[\mathcal{Q} \mid \mathcal{E}]}{\partial \Pr[\mathcal{B} \mid \mathcal{A}]}$ by applying elementary calculus.

Relation to Other Inference Methods

Bucket elimination is one method for carrying out belief network inference (as well as a number of other inferential processes in other domains). There are other approaches, for example *junction-tree* algorithms (see Cowell et al 1999), which compile the belief network into a representation whose space-complexity is potentially much greater, but for which inference is potentially much faster. We have not looked in detail at the problem, but believe it is possible to apply our method in this context as well.

Recent work by Darwiche (2000) describes a method for compiling a belief network into a polynomial representation, which may then be symbolically differentiated. There appears to be a close similarity between his approach and ours, as the polynomial can be thought of as a symbolic representation of the variable elimination process, and the differentiation proceeds from the leaves of the tree-representation of the polynomial to its root, tracing out the same pattern in space as our algorithm traces out in time. We need to compare the approaches more closely to determine the exact relationship between them, however.

5.2.4 Algorithm

Here we incorporate the results of this section into an algorithm for belief network inference:

1. Compute the response, $q(\bar{\theta})$.
2. Compute the first derivative, $q'(\bar{\theta})$.
3. Compute the error-bars around $q(\bar{\theta})$:
   (a) Let var = 0.
   (b) For each row of parameters, $\langle \bar{\theta}_r \ldots \bar{\theta}_s \rangle$, do:
      i. Compute the variance-covariance matrix:
$$\begin{bmatrix} \sigma_{rr} & \cdots & \sigma_{rs} \\ \vdots & \ddots & \vdots \\ \sigma_{sr} & \cdots & \sigma_{ss} \end{bmatrix}$$
      using the appropriate formula for their joint distribution — see Appendix A.
      ii. Increment var by:
$$\sum_{j=r}^{s} \left[\sum_{i=r}^{s} q'_i(\bar{\theta}) \cdot \sigma_{ij} \right] \cdot q'_j(\bar{\theta})$$
      (matrix multiplication).
   (c) Let $\varepsilon = -\sqrt{\text{var}} \cdot \Phi^{-1}(\delta/2)$, where $\Phi^{-1}(\cdot)$ is the inverse of the standard normal distribution function, and δ is the desired confidence/ credibility level.
4. Return $q(\bar{\theta}) \pm \varepsilon$.

5.3 Empirical Study

We carried out a number of experiments in order to assess the viability of our method. We wanted to know whether or not the approximations used were reasonable, and whether it would scale up to non-trivial problems. Therefore we carried out a series of experiments to evaluate the accuracy of our approximate credible intervals.

We used the following test domains:

- The diamond graph: a network of manageable complexity with a variety of possible inference patterns. See Figure 3.1.
- The Alarm network (Bienlich et al 1989) a benchmark network commonly used in belief network studies, based on a medical diagnosis domain. Subsets of the network variables are identified as query variables or evidence variables. Variables are all discrete, but most range over 3 or more values.
- Random networks: We used networks on 10 binary variables, with 20 links drawn at random and probability vectors (in the gold model — see below) drawn from uniform Dirichlet distributions.

5.3.1 Experimental Design

Our experiments consisted of carrying out a set of inferences on each network and evaluating the resulting $(1-\delta)$ credible intervals by generating a sample of values distributed according to the posterior for the query. Using this Monte Carlo method, we obtained in each case an empirical estimate $\hat{\delta}$ of the actual amount of probability mass under the posterior that lay outside the interval. 100 inferences were used to generate $\hat{\delta}$ in each of our experiments. $|\hat{\delta}-\delta|$ is a measure of how accurate an error-bar is — it is the error in the confidence/credibility level claimed for the error-bars.

Below we summarize our results on three domains: (1) the 4-node diamond graph, (2) the Alarm network, and (3) random networks on 10 binary variables, with 20 links between them. We looked at the effects of varying the credibility level, δ, the sample size used for training, m, and the nature of the queries.

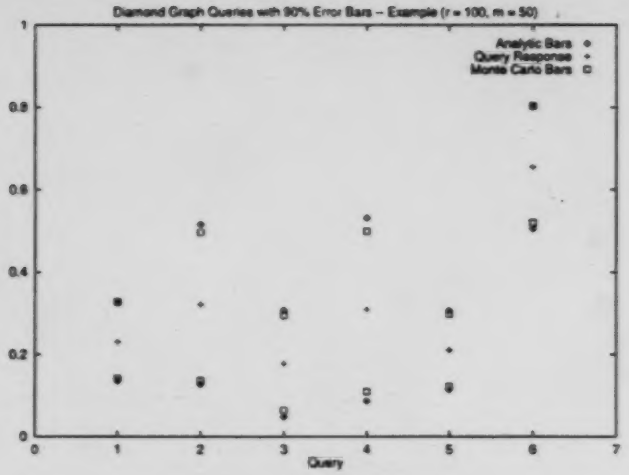


Figure 5.1: Examples of Error Bars

5.3.2 Results for the Diamond Graph

We studied the following inferential patterns in the diamond graph:

1. $\Pr[\mathcal{X}_1]$
2. $\Pr[\mathcal{X}_1 \mid \mathcal{X}_2]$
3. $\Pr[\mathcal{X}_1 \mid \mathcal{X}_2, \mathcal{X}_3]$
4. $\Pr[\mathcal{X}_2, \mathcal{X}_3 \mid \mathcal{X}_1]$
5. $\Pr[\mathcal{X}_1 \mid \mathcal{X}_4]$
6. $\Pr[\mathcal{X}_4 \mid \mathcal{X}_1]$

We carried out many trials of the following form: (1) generate a "true" model, by generating probabilities for the network's CP-tables, (2) generate a posterior distribution over the parameters, by generating a sample of size m from the true model, and (3) evaluate the above inferences on the resulting network.

Figure 5.1 shows the error-bars returned by each method on a random network posterior. Here the error-bars are 90% credible intervals, and close agreement is seen between the two methods.

Figure 5.2 compares an empirical distribution around a query response to its analytically derived counterpart. The stepped function represents the results of a Monte Carlo simulation; the smooth curve is a normal distribution with the empirical variance. This tests the validity of the normality assumption *independently* of the linear approximation. The functions are based on an empirical distribution of *error* (distance from the mean of the distribution of the query response), generated by inferences on 100 models drawn from the network posterior. This is for a single, randomly chosen posterior on the diamond graph, and the inference is Query 6 in the list above. (Note the normal distribution is scaled by the unit length chosen for the histogram, so there is equal area under the two distributions.)

Table 5.1 summarizes the results of a comprehensive study on the diamond network. Each inner cell contains an average of 30 $|\hat{\delta} - \delta|$ values, generated from experiments of the form described previously. The outer cells are marginalizations over rows, columns, and the entire table.

Table 5.1: Results for the Diamond Graph

$\delta = 0.1$

	Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	average
$m = 10$	0.023667	0.027667	0.031000	0.032667	0.022000	0.039333	0.029389
$m = 20$	0.026667	0.033333	0.025000	0.033667	0.026000	0.035000	0.029944
$m = 30$	0.026000	0.030333	0.024000	0.031333	0.029000	0.037000	0.029611
$m = 40$	0.026000	0.029667	0.031000	0.020000	0.027667	0.029000	0.027222
average	0.025583	0.030250	0.027750	0.029417	0.026167	0.035083	0.029042

$\delta = 0.2$

	Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	average
$m = 10$	0.035000	0.050000	0.047000	0.048667	0.034333	0.055667	0.045111
$m = 20$	0.046000	0.062667	0.051333	0.070333	0.040333	0.045333	0.052667
$m = 30$	0.029000	0.050667	0.044333	0.059667	0.039667	0.048667	0.045333
$m = 40$	0.040667	0.052667	0.049333	0.049333	0.040333	0.038667	0.045167
average	0.037667	0.054000	0.048000	0.057000	0.038667	0.047083	0.047069

$\delta = 0.3$

	Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	average
$m = 10$	0.036333	0.056333	0.061000	0.072333	0.051333	0.062667	0.056667
$m = 20$	0.047000	0.072000	0.068333	0.111333	0.053000	0.060333	0.068667
$m = 30$	0.037000	0.058000	0.054667	0.072000	0.035000	0.053000	0.051611
$m = 40$	0.051333	0.069667	0.060333	0.066333	0.042667	0.042667	0.055500
average	0.042917	0.064000	0.061083	0.080500	0.045500	0.054667	0.058111

$\delta = 0.4$

	Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	average
$m = 10$	0.043333	0.039667	0.053333	0.075333	0.044000	0.066333	0.053667
$m = 20$	0.047333	0.052000	0.057333	0.092667	0.052000	0.059667	0.060167
$m = 30$	0.033333	0.045000	0.050000	0.064667	0.042667	0.049667	0.047556
$m = 40$	0.039000	0.049000	0.059667	0.067333	0.044333	0.047000	0.051056
average	0.040750	0.046417	0.055083	0.075000	0.045750	0.055667	0.053111

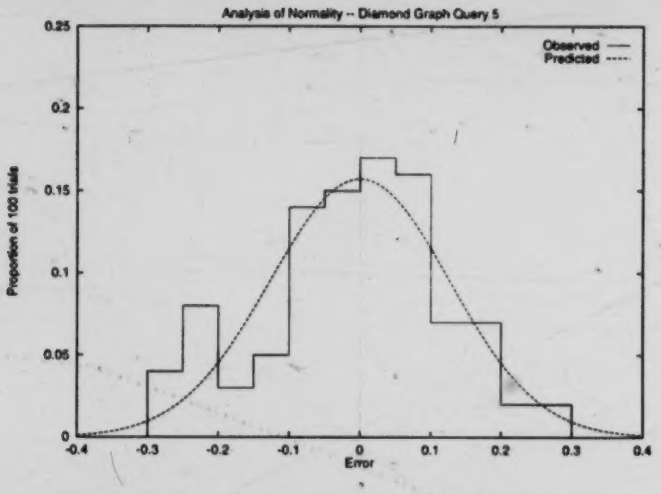


Figure 5.2: Testing the validity of the normal approximation.

Table 5.2: Results for the Alarm Network

	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.4$	average
$m = 50$	0.024700	0.043700	0.044800	0.040700	0.038475
$m = 100$	0.026600	0.049500	0.059700	0.048700	0.046125
$m = 150$	0.030400	0.053500	0.064500	0.056600	0.051250
$m = 200$	0.026500	0.048000	0.054300	0.054200	0.045750
average	0.027050	0.048675	0.055825	0.050050	0.045400

5.3.3 Results for Alarm

Table 5.2 summarizes the results for experiments on the Alarm network, where both δ and m were varied. Each inference was of the form $\Pr[\mathcal{Q} \mid \mathcal{E}]$, where $\mathcal{Q}$ was an assignment to a *single* randomly chosen query variable (from the set of variables identified as such) and $\mathcal{E}$ was an assignment to 5 randomly chosen evidence variables (again, from a prespecified set). Some (or all) of the evidence variables might have had no effect on the query variable, others might have had a profound effect. Each inner cell summarizes 100 queries on a single posterior distribution.

5.3.4 Results for Random Networks

Although random networks tend not to reflect typical (or natural) domains, they complement more focussed studies by exposing methods to a wide range of inputs and help to support claims of generality. We carried out experiments on networks with 10 binary variables and 20 links, generating gold models from a uniform distribution over model parameters, and generating random queries of various types. Here we used $m = 100$ throughout, and varied the type of query. Table 5.3 displays the results of our experiments. Each query was of the form $\Pr[\mathcal{Q} \mid \mathcal{E}]$, and $\#\mathcal{E}$ denotes the number of variables assigned a value in $\mathcal{E}$; likewise for $\#\mathcal{Q}$. Each inner cell is based on 100 trials: 10 queries on 10 networks (with both structure and posterior generated randomly.

Table 5.3: Results for Random Graphs

$\delta = 0.1$

	$\#\mathcal{E}=1$	$\#\mathcal{E}=2$	$\#\mathcal{E}=3$	$\#\mathcal{E}=4$	$\#\mathcal{E}=5$	average
$\#\mathcal{Q}=1$	0.021600	0.027200	0.028400	0.032000	0.030000	0.027840
$\#\mathcal{Q}=2$	0.025900	0.025000	0.029300	0.031300	0.024500	0.027200
$\#\mathcal{Q}=3$	0.024900	0.025900	0.026200	0.023800	0.025600	0.025280
$\#\mathcal{Q}=4$	0.025700	0.022600	0.028400	0.025800	0.028800	0.026260
$\#\mathcal{Q}=5$	0.027200	0.025300	0.026100	0.030500	0.027900	0.027400
average	0.025060	0.025200	0.027680	0.028680	0.027360	0.026796

$\delta = 0.2$

	$\#\mathcal{E}=1$	$\#\mathcal{E}=2$	$\#\mathcal{E}=3$	$\#\mathcal{E}=4$	$\#\mathcal{E}=5$	average
$\#\mathcal{Q}=1$	0.030600	0.041600	0.044100	0.041900	0.043900	0.040420
$\#\mathcal{Q}=2$	0.036000	0.036300	0.043800	0.047100	0.049600	0.042560
$\#\mathcal{Q}=3$	0.035000	0.041600	0.045600	0.050700	0.057800	0.046140
$\#\mathcal{Q}=4$	0.041200	0.044600	0.056400	0.063100	0.071400	0.055340
$\#\mathcal{Q}=5$	0.046700	0.057600	0.066300	0.078500	0.081700	0.066160
average	0.037900	0.044340	0.051240	0.056260	0.060880	0.050124

$\delta = 0.3$

	$\#\mathcal{E}=1$	$\#\mathcal{E}=2$	$\#\mathcal{E}=3$	$\#\mathcal{E}=4$	$\#\mathcal{E}=5$	average
$\#\mathcal{Q}=1$	0.041700	0.047800	0.051100	0.047500	0.059700	0.049560
$\#\mathcal{Q}=2$	0.044600	0.042800	0.050200	0.051300	0.063300	0.050440
$\#\mathcal{Q}=3$	0.041200	0.047400	0.051400	0.063900	0.072500	0.055280
$\#\mathcal{Q}=4$	0.049800	0.056100	0.074300	0.087100	0.108100	0.075080
$\#\mathcal{Q}=5$	0.056900	0.076300	0.094500	0.125100	0.139900	0.098540
average	0.046840	0.054080	0.064300	0.074980	0.088700	0.065780

$\delta = 0.4$

	$\#\mathcal{E}=1$	$\#\mathcal{E}=2$	$\#\mathcal{E}=3$	$\#\mathcal{E}=4$	$\#\mathcal{E}=5$	average
$\#\mathcal{Q}=1$	0.042000	0.049000	0.050000	0.045600	0.056200	0.048560
$\#\mathcal{Q}=2$	0.044300	0.042400	0.048100	0.054400	0.064300	0.050700
$\#\mathcal{Q}=3$	0.041100	0.046200	0.048000	0.059500	0.071200	0.053200
$\#\mathcal{Q}=4$	0.051500	0.045200	0.064700	0.083600	0.109900	0.070980
$\#\mathcal{Q}=5$	0.049500	0.071400	0.089500	0.130500	0.161500	0.100480
average	0.045680	0.050840	0.060060	0.074720	0.092620	0.064784

5.3.5 Discussion

Our hypothesis was that our error-bars would be accurate for a large number of cases. We tried to falsify our hypothesis by varying the following experimental parameters:

- Network structure.
- Credibility level, δ.
- Query type (Diamond network).
- Number of evidence variables (Random networks).
- Number of query variables (Random networks).

In no case did we observe a result where the average $|\hat{\delta} - \delta|$ exceeded 0.2. In most cases, the error was less than $\delta/3$. Note that, even if our error-bars were exact, we would get non-zero results due to the variance in the simulation process. Therefore we believe that these results comfortably bound the expected error of our method under the experimental conditions. None of the variables that we manipulated had a profound effect. The strongest effect observed was that increasing the number of variables assigned in a query tended to increase the error in $\hat{\delta}$. One possible explanation is that, by increasing the number of variables assigned, we tend to *decrease* the number of parameters involved in computing the answer. This could make the Normality approximation less accurate, as it depends on a limiting result for the sum of the individual parameter errors. Another possibility is that, by increasing the number of assigned variables, the answer probability tended to become very small, and thus floating point imprecision became more of an issue. Further experiments could address this issue.

We found these results very encouraging — our method appears to give reasonable error-bars for a wide range of queries and network types. This makes it a promising technique to add to the array of data-analysis tools related to belief networks, especially as it is reasonably efficient, roughly doubling the computation time per inference. There must be pathological cases where our method will not give reasonable results, when the local linear-normal assumption is far off the truth (although we did not find any in our experiments). Nevertheless, error-bars are an important source of information about the underlying distribution around a point estimate, and even approximate error-bars such as these can provide valuable guidance for decision-making or learning.

Chapter 6

Conclusion

In this dissertation we explored issues that arise because of the uncertainty in estimates of belief network parameters. This uncertainty is represented by a posterior distribution over parameters (Bayesian perspective) or an unknown sampling distribution (Frequentist perspective). The uncertainty in parameter values adds a new level of uncertainty to the uncertainty already expressed by the probabilistic model itself. We explored the implications for learning network structure and for making inferences based on network parameters.

We looked at the problem of model selection, where the major issue is how to handle the bias-variance trade-off: more complex models have a lower bias but more variance in their parameters. We presented a theoretical and empirical comparison of how various model selection criteria operationalize this trade-off. There is no "first-principles" justification for one criterion over another, as we have shown. The choice of a model selection criterion, therefore, ought to reflect prior knowledge and pragmatic concerns. Prior knowledge and pragmatic concerns are often a matter of intuition, and it can be difficult to capture those intuitions formally, as a scoring function. It is therefore of interest to compare criteria across a variety of domains, to see how they behave in practice, in order to decide which criterion to apply to a particular learning problem. To this end we carried out an empirical comparison of model selection criteria.

We found that a minimum description length criterion was slow to converge, and thus tended to underfit. Akaike's information criterion performed reasonably well as an estimator of true error, except on complex true distributions or small samples, where its slow rate of convergence also led to underfitting. Both the AIC and MDL criteria are based on asymptotics, and thus convergence rates are critical to their performance. A bootstrap criterion led to overfitting in most cases, as it seemed to reflect sample error as much or more than true error. The prequential criterion and 2-fold cross-validation performed reasonably well on all the problem spaces we considered, being good predictors of true error. Results such as these, complemented by an understanding of the theoretical issues involved, can provide guidance in making the decision to use a particular criterion for a particular learning problem.

We also looked at the uncertainty in inferences based on belief network parameters. We derived approximate error-bars for belief network inferences, based on a linear approximation to the error of an inference, and a Normal approximation to its distribution. By propagating the variances and covariances of parameters through their partial derivatives with respect to the inference, we were able to derive an approximate variance for the error of a query, and thus provide a credible interval for it. In this derivation, we relied heavily on the conjugate property of belief networks with Dirichlet priors (Theorem 1) to make the analysis tractable. A major difficulty in computing the error-bars was the subproblem of computing the partial derivatives of network parameters with respect to an inference. We gave an efficient algorithm for solving this subproblem, based on the belief network inference procedure known as bucket elimination. Last, we presented the results of an empirical

study showing that the approximations involved in our method are reasonable and scale up to non-trivial problems.

Bayesian methods figured prominently in our research. In the area of model selection, we saw how using the mean parameter vector (the Bayesian predictive distribution) rather than the maximum-likelihood parameters made the goodness of fit a better predictor of true error. As well, one of the more elegant model selection criteria, the prequential criterion, has a plausible interpretation as a Bayesian method. For error-bars, credible regions turned out to be much easier to derive than confidence regions. In both cases, the advantage of Bayesian methods resulted from the conjugate property of Dirichlet priors, and the ease of integrating out the parameters under the posterior.

There remain some areas for future work. In particular, we considered only belief networks over discrete-valued variables with Dirichlet priors. There is a broader class of graphical models to which some or all of these results may be applicable. We are interested in extending this work to other kinds of networks. There is the also the issue of "incomplete" data — our analyses assumed that the sample was a collection of complete assignments to all network variables. In practice, there are cases where some values are absent, or where one is given several heterogenous databases to learn from. Different methods for parameter estimation are required for these situations, with consequences for theoretical analysis.

Handling uncertainty when you're handling uncertainty is a fundamental problem in machine learning. We often have a need to represent and reason with uncertain knowledge, and it is often the case that such uncertain knowledge is itself the outcome of a learning process which imparts an additional level of uncertainty. Belief networks are a general representation scheme for uncertain knowledge about stationary domains, and belief network inference is a general method for inference about such domains. Choosing a network structure is a difficult problem in learning a belief network from data — it involves trade-offs between simplicity and generalizeability, bias and variance. Once a structure is chosen and parameters are estimated, a belief network is used to do probabilistic inference. Here a difficult problem is propagating the uncertainty in the parameter values during the inference process. These are the two problems that we have addressed. Our results provide guidance for making the trade-offs of model selection, and provide a method for giving approximate error-bars for inferences. Together, we believe these results are an important addition to the science of learning and reasoning with belief networks.

References

Bishop, Y., Fienberg, S. & Holland, P. (1995). *Discrete Multivariate Analysis — Theory and Practice*, MIT Press.

Bozdogan, H. (1987). Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions. *Psychometrika, 52, 3*, 345–370.

Castillo, E., Gutierrez, J., & Hadi, A. (1997). Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Man, Cybernetics and Systems, 27*, 412–424.

Che, P., Neapolitan, R., Kenevan, J., & Evens, M. (1993). An implementation of a method for computing the uncertainty in inferred probabilities in belief networks. *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann.

Cooper, G., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9*, 309–347.

Cover, T. & Thomas, J. (1991). *Elements of Information Theory*. John Wiley & Sons.

Cowell, R., Dawid, A., Lauritzen, S. & Spiegelhalter, D. (1999). *Probabilistic Networks and Expert Systems*. Springer.

Darwiche, A. (2000). A differential approach to inference in Bayesian networks. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Dawid, P. & Lauritzen, S. (1993). Hyper markov laws in the statistical analysis of decomposable graphical models, *Annals of Statistics*, 1993.

Dawid, P. & Vovk, V. (1999). Prequential probability: Principles and properties. *Bernoulli 5*, 125-162.

Dechter, R. (1996). Bucket elimination: A unifying framework for probabilistic inference. *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 211–219.

Efron, B. (1982). *The Jackknife, the Bootstrap and other Resampling Plans*. Society for Industrial and Applied Mathematics, Philadelphia.

Friedman, N., & Goldszmidt, M. (1996a). Learning Bayesian networks with local structure. *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann.

Friedman, N., & Yakhini, Z. (1996b). On the sample complexity of learning Bayesian networks. *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann.

Heckerman, D. (1995). *A tutorial on learning with Bayesian networks* Technical Report MSR-TR-95-06, Microsoft Research.

Heckerman, D. (1996). A comparison of scientific and engineering criteria for Bayesian model selection. Technical Report MSR-TR-96-12, Microsoft Research.

Howson, C. (1997). A logic of induction. *Philosophy of Science*, 64, (pp. 268-290).

Kearns, M., Mansour, Y., Ng, A. Y., & Ron, D. (1997). An experimental and theoretical comparison of model selection methods. *Machine Learning*, 14.

Kleiter, G. (1996). Propagating imprecise probabilities in Bayesian networks. *Artificial Intelligence 88*, 143-161.

Korth, H., Silberschatz, A. & Sudarshan, S. (1998). *Database System Concepts*. McGraw Hill.

Kullback, S. & Leibler, R. (1951). On information and sufficiency. *Annals of Mathematical Statistics 22*, 79–86.

Lam, W., & Bacchus, F. (1994). Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence, 10*, 269–293.

Laskey, K. (1995). Sensitivity analysis for probability assessments in Bayesian networks, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 6.

Linhart, H., & Zucchini, W. (1986). *Model selection*. New York: John Wiley & Sons.

Liu, J., Chang, K. & Zhou, J. (1998). A hybrid convergent method for learning probabilistic networks. *Proceedings of the 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, 393–410.

Newman, T. & Odell, P. (1971). *The Generation of Random Variates*, Vol. 21 of Griffen's Statistical Monographs and Courses, Griffen, London, 1971.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Rissanen, J. (1989). *Stochastic complexity in statistical inquiry*. World Scientific.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics, 6*, 461–464.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *J. Royal Statistical Society, Series B, 36*, 44-47.

Stone, M. (1977). Asymptotics for and against cross-validation. *Biometrika, 64, 1*, 29-35.

Suzuki, J. (1996). Learning bayesian belief networks based on the minimum description length principle: An efficient algorithm using the branch and bound technique. *Machine Learning 13*.

Wilks, S. (1962). *Mathematical Statistics*, John Wiley & Sons, New York.

Wolpert, D. (1996a). The lack of a priori distinctions between learning algorithms. Available at `ftp://ftp.santafe.edu/pub/dhw_ftp/`.

Wolpert, D. (1996b). The existence of a priori distinctions between learning algorithms. Available at `ftp://ftp.santafe.edu/pub/dhw_ftp/`.

Appendix A

Distributions Used

The properties of the distributions given in this Appendix are described in most intermediate level texts on Mathematical Statistics. A good reference which we use is Wilks (1962). The code that generates random variates is either based on obvious properties of the distributions, or comes from Newman and Odell (1971).

The Multivariate Bernoulli Distribution

Suppose we have n discrete variables $X_1 \ldots X_n$, and $\mathbf{X}_i = \{0, 1\}$ for $i = 1$ to n. An $(n-1)$-variate Bernoulli distribution on these variables is given by:

$$\text{Ber}(X_1 \ldots X_n \mid \theta_1 \ldots \theta_n) = \prod_{i=1}^{n} \theta_i^{X_i}$$

subject to the constraints: $\theta_1 + \cdots + \theta_n = 1$, $X_1 + \cdots + X_n = 1$ and $\theta_i \geq 0$ for $i = 1$ to n. Note that this is an $(n-1)$-variate distribution because the variables are constrained by a linear equation that defines an $n-1$ dimensional surface in $\Re^n$. The mean of this distribution is $\theta = \langle \theta_1 \ldots \theta_n \rangle$, and the variance-covariance matrix is given by:

$$\sigma_{ij} = \begin{cases} \theta_i(1-\theta_i) & \text{if } i = j \\ -\theta_i\theta_j & \text{otherwise} \end{cases}$$

To generate a random value from this distribution, carry out the following algorithm:

```
Random_Bernoulli(n, theta) =
BEGIN
    x = NEW(VECTOR, n)
    p = Random_Uniform(0.0, 1.0)
    q = 0.0
    FOR i = 1 TO n DO
        IF (q < p AND p <= q + theta[i])
        OR (q <= p AND p < q + theta[i]) THEN
            x[i] = 1
        ELSE
            x[i] := 0
        END
        q := q + theta[i]
    END
    RETURN x
END
```

The Binomial and Multinomial Distributions

The $(n-1)$-variate Multinomial distribution is a distribution on the discrete variables $A_1 \ldots A_n$, where the vector A is the vector sum of m instances of $\langle X_1 \ldots X_n \rangle$, drawn from an $(n-1)$-variate Bernoulli distribution with parameters $\theta_1 \ldots \theta_{n-1}$. It is given by:

$$\text{Mult}(A_1 \ldots A_n \mid m, \theta_1 \ldots \theta_n) = \frac{m!}{A_1! \cdots A_n!} {\theta_1}^{A_1} \cdots {\theta_n}^{A_n}$$

subject to the constraints: $\theta_1 + \cdots + \theta_n = 1$, $A_1 + \cdots + A_n = m$ and $A_i \geq 0, \theta_i \geq 0$ for $i = 1$ to n. The expectation is $\langle m\theta_1 \ldots m\theta_n \rangle$, and the variance-covariance matrix is given by:

$$\sigma_{ij} = \begin{cases} m\theta_i(1-\theta_i) & \text{for } i = j \\ -m\theta_i\theta_j & \text{otherwise} \end{cases}$$

The Binomial is the special case of the Multinomial for $n = 2$. The following procedure returns a random value from a Multinomial:

```
Random_Multinomial(n, theta, m) =
BEGIN
    A = NEW(VECTOR, n)
    FOR i = 1 TO n DO
        A[i] = 0
    END
    FOR i = 1 TO m DO
        x = Random_Bernoulli(n, theta)
        FOR j = 1 TO n DO
            A[i] := A[i] + x[i]
        END
    END
    RETURN A
END
```

The Gamma Distribution

The Gamma distribution is described here because of its relation to the Beta and Dirichlet distributions, which follow. It is a distribution on a continuous variable X whose domain is $\mathbf{X} = (0, \infty)$, the positive real numbers. It is given by:

$$\text{Ga}(X \mid \lambda, \mu) = \Gamma(\mu)^{-1} (\lambda X)^{\mu - 1} e^{-\lambda X}$$

where the Gamma function is defined as follows:

$$\Gamma(\mu) = \int_0^\infty X^{\mu-1} e^{-X} \, dX$$

Note that $\Gamma(\mu) = (\mu - 1)\Gamma(\mu - 1)$. Thus, when μ is a positive integer $\Gamma(\mu) = (\mu - 1)!$. The expectation and variance of the Gamma distribution are both equal to μ when $\lambda = 1$. The following procedure generates a random value from a Gamma distribution when μ is an integer:

```
Random_Gamma(lambda, mu) =
BEGIN
    sum = 0.0
    FOR i = 1 TO mu DO
        sum := sum - ln(Random_Uniform(0, 1))
    END
    RETURN sum / lambda
END
```

The Beta and Dirichlet Distributions

The $(n-1)$-variate Dirichlet is a distribution on the $n-1$ parameters of an $(n-1)$-variate Bernoulli distribution; the Beta distribution corresponds to the special case where $n = 2$. It is given by:

$$\mathrm{Dir}(\Theta_1 \ldots \Theta_n \mid \alpha_1 \ldots \alpha_n) = \frac{{\Theta_1}^{\alpha_1 - 1} \ldots {\Theta_n}^{\alpha_n - 1}}{D(\alpha_1 \ldots \alpha_n)}$$

The Beta distribution corresponds to the special case where $n = 2$. $D(\alpha_1 \ldots \alpha_n)$ is the Dirichlet integral, given by:

$$\begin{aligned} D(\alpha_1 \ldots \alpha_n) &= \int \ldots \int {\Theta_1}^{\alpha_1 - 1} \ldots {\Theta_n}^{\alpha_n - 1} \, d\Theta_1 \cdots d\Theta_n \\ &= \frac{\Gamma(\alpha_1) \cdots \Gamma(\alpha_n)}{\Gamma(\alpha_1 + \cdots + \alpha_n)} \end{aligned}$$

The mean vector of a Dirichlet distribution is given by $\bar{\theta}_i = \alpha_i / (\alpha_1 + \cdots + \alpha_n)$, the mode vector is given by $\hat{\theta}_i = (\alpha_i - 1)/(\alpha_1 + \cdots \alpha_n - n)$, where it exists, and the variance-covariance matrix is given by:

$$\sigma_{ij} = \begin{cases} \frac{\alpha_i(\alpha_1 + \cdots + \alpha_n - \alpha_i)}{(\alpha_1 + \cdots + \alpha_n)^2(\alpha_1 + \cdots + \alpha_n + 1)} & \text{if } i = j \\ \frac{-\alpha_i \alpha_j}{(\alpha_1 + \cdots + \alpha_n)^2(\alpha_1 + \cdots + \alpha_n + 1)} & \text{otherwise} \end{cases}$$

The following procedure generates a random value from a Dirichlet distribution when the α_i parameters are all positive integers (which is the case for many applications):

```
Random_Dirichlet(n, alpha) =
BEGIN
    x = NEW(VECTOR, n)
    y = NEW(VECTOR, n)
    sum = 0.0
    FOR i = 1 TO n DO
        x[i] := Random_Gamma(1.0, alpha[i])
        sum := sum + x[i]
    END
    FOR i = 1 TO n DO
        y[i] := x[i] / sum
    END
    RETURN y
END
```

Appendix B

Properties of $f(\Theta \mid a + \alpha)$

Here we demonstrate some important properties of the posterior distribution $f(\Theta \mid a + \alpha)$, which follows Bayesian updating of the prior distribution. In particular, we claim that, for a fixed network structure, when the prior over the parameter space is given by a product of Dirichlets, one for the parameters of each CP-table row (each row is the conditional multivariate Bernoulli distribution of a node given its parents), then the following hold:

- Theorem 1 (Conjugate Property): The posterior has the same form as the prior and its parameters are given by updating the individual Dirichlet distributions. In other words, each row of a CP-table is independently Dirichlet-distributed in the posterior distribution, as it is in the prior.
- Theorem 2 (Model Averaging): The predictive distribution, which involves averaging over all models according to their distribution, is the model which is the mean of the distribution over parameter space.
- Corollary of Theorem 2 (Prequential Likelihood): The form of the predictive likelihood, which also involves marginalizing out the parameters of the network, but defines a distribution over all samples of size m, is given by the product of the probabilities of each datum, given the posterior for the data that preceded it in the sample (order is irrelevant).

We prove these properties here for the special case where all network variables are binary, and CP-tables are complete; the generalization to n-ary variables and marginalized CP-tables is conceptually straightforward, but the notation becomes cumbersome. See also Cooper and Herskovits (1992) who make these claims without proof (their proofs are available in a technical report). Dawid and Lauritzen (1993) present more general results. We present these results here to make the paper self-contained.

First we derive the likelihood function $L(d \mid m, \Theta)$. Because d is a sequence $d_1 \ldots d_m$ of i.i.d. instances of $X_1 \ldots X_n$, we can write:

$$L(d \mid m, \Theta) = \prod_{i=1}^{m} P(d_i \mid \Theta)$$

And, because we interpret our model as a conditional factorization of a joint probability distribution, we can write:

$$P(d_i \mid \Theta) = \prod_{j=1}^{n} P_j(x_{ij} \mid w_{ij})$$

where x_{ij} is the assignment to X_j in d_i, and w_{ij} is the joint assignment to W_j in d_i.

Since $P_j(x_{ij} \mid w_{ij})$ is a network parameter, or pseudo-parameter, we can rearrange the factors as:

$$L(d \mid m, \Theta) = \prod_{i=1}^{k} \Theta_i^{a_i}$$

where a_i (as previously defined) is the observed frequency in the sufficient statistic for Θ.

Next we derive the posterior density function for Θ. Baye's theorem gives us:

$$f(\Theta \mid a + \alpha) = \frac{L(d \mid m, \Theta) f(\Theta \mid \alpha)}{\int L(d \mid m, \Theta) f(\Theta \mid \alpha)\, d\Theta}$$

We can take the parameters to be jointly Beta-distributed in pairs, ($\Theta_{i+1} = 1 - \Theta_i$ when i is odd) and given our previous equality for the likelihood function, write the posterior as:

$$f(\Theta \mid a + \alpha) = \frac{\Theta_1^{a_1}\Theta_2^{a_2}\mathrm{Be}(\Theta_1 \mid \alpha_1, \alpha_2)\cdots\Theta_{k-1}^{a_{k-1}}\Theta_k^{a_k}\mathrm{Be}(\Theta_{k-1} \mid \alpha_{k-1}, \alpha_k)}{\int\cdots\int \Theta_1^{a_1}\Theta_2^{a_2}\mathrm{Be}(\Theta_1 \mid \alpha_1, \alpha_2)\cdots\Theta_{k-1}^{a_{k-1}}\Theta_k^{a_k}\mathrm{Be}(\Theta_{k-1} \mid \alpha_{k-1}, \alpha_k)\, d\Theta_1 \cdots d\Theta_{k-1}}$$

Using the definition of the Beta distribution, and our assumptions about the form of Θ, for $i = 1$ to k:

$$\begin{aligned} \Theta_i^{a_i}\Theta_{i+1}^{a_{i+1}}\mathrm{Be}(\Theta_i \mid \alpha_i, \alpha_{i+1}) &= \Theta_i^{a_i}(1-\Theta_i)^{a_{i+1}}\Theta_i^{\alpha_i - 1}(1-\Theta_i)^{\alpha_{i+1}-1}D(\alpha_i, \alpha_{i+1}) \\ &= \Theta_i^{a_i+\alpha_i-1}(1-\Theta_i)^{a_{i+1}+\alpha_{i+1}-1}D(\alpha_i, \alpha_{i+1}) \end{aligned}$$

We can then cancel the Dirichlet integrals in the top with those on the bottom, and factor out the integration to get:

$$f(\Theta \mid a + \alpha) = \frac{\Theta_1^{a_1+\alpha_1-1}(1-\Theta_1)^{a_2+\alpha_2-1}\cdots\Theta_{k-1}^{a_{k-1}+\alpha_{k-1}-1}(1-\Theta_{k-1})^{a_k+\alpha_k-1}}{\int_0^1 \Theta_1^{a_1+\alpha_1-1}(1-\Theta_1)^{a_2+\alpha_2-1}\, d\Theta_1 \cdots \int_0^1 \Theta_{k-1}^{a_{k-1}+\alpha_{k-1}-1}(1-\Theta_{k-1})^{a_k+\alpha_k-1}\, d\Theta_{k-1}}$$

and since the integrals in the denominator are equal to Beta functions (Dirichlet integrals with two parameters) it follows that:

$$f(\Theta \mid a + \alpha) = \mathrm{Be}(\Theta_1 \mid a_1 + \alpha_1, a_2 + \alpha_2)\cdots\mathrm{Be}(\Theta_{k-1} \mid a_{k-1} + \alpha_{k-1}, a_k + \alpha_k)$$

a product of independent Beta distributions, one for each row of a CP-table $\langle \Theta_i, \Theta_{i+1} \rangle$ (for i odd), as claimed. Furthermore, the posterior parameters are updated by just adding the statistic a to the prior parameters, α. This proves the binary variable version of Theorem 1.

Theorem 2 follows directly. The predictive distribution is given by:

$$Q(X \mid \alpha) = \int P(X \mid \Theta) f(\Theta \mid \alpha)\, d\Theta$$

where α is the current parameter of the distribution over parameter space, regardless of whether it is conditioned on observed data or not. Our claim is that:

$$Q(X \mid \alpha) = P(X \mid \bar{\theta})$$

where $\bar{\theta}$ is the mean of the posterior distribution, given by $\bar{\theta}_i = \alpha_i/(\alpha_i + \beta_i)$ for $i = 1$ to k. Note that, for a particular full instantiation $x_1 \ldots x_n$, $P(x_1 \ldots x_n \mid \Theta)$ depends on at most one parameter per CP-table row. So the predictive distribution is given by a product of n independent factors of the form:

$$\int \Theta_i \mathrm{Be}(\Theta_i \mid \alpha_i, \beta_i)\, d\Theta_i = \int \frac{\Theta_i^{\alpha_i}(1-\Theta_i)^{\beta_i - 1}}{D(\alpha_i, \beta_i)}\, d\Theta_i = \frac{D(\alpha_i + 1, \beta_i)}{D(\alpha_i, \beta_i)}$$

$$= \frac{\Gamma(\alpha_i + 1)\Gamma(\beta_i)\Gamma(\alpha_i + \beta_i)}{\Gamma(\alpha_i + \beta_i + 1)\Gamma(\alpha_i)\Gamma(\beta_i)} = \frac{(\alpha_i)\Gamma(\alpha_i)\Gamma(\alpha_i + \beta_i)}{(\alpha_i + \beta_i)\Gamma(\alpha_i + \beta_i)\Gamma(\alpha_i)} = \frac{\alpha_i}{\alpha_i + \beta_i}$$

as claimed.

A corollary of this is that the predictive likelihood, $\mathrm{PL}(d_1 \ldots d_m \mid m, \alpha)$, is given by a prequential algorithm:

$$\begin{aligned}
\mathrm{PL}(d \mid m, \alpha) &= \int L(d \mid m, \Theta) f(\Theta \mid \alpha)\, \mathrm{d}\Theta \\
&= \int \prod_{i=1}^{m} P(d_i \mid \Theta) f(\Theta \mid \alpha)\, \mathrm{d}\Theta \\
&= \int \prod_{i=1}^{k} \Theta_i{}^{a_i} f(\Theta \mid \alpha)\, \mathrm{d}\Theta \\
&= \int \Theta_1{}^{a_1} (1-\Theta_1)^{b_1} \mathrm{Be}(\Theta_1 \mid \alpha_1, \beta_1)\, \mathrm{d}\Theta_1 \\
&\quad \vdots \\
&\quad \int \Theta_{k-1}{}^{a_{k-1}} (1-\Theta_{k-1})^{b_{k-1}} \mathrm{Be}(\Theta_{k-1} \mid \alpha_{k-1}, \beta_{k-1})\, \mathrm{d}\Theta_{k-1}
\end{aligned}$$

where a and b are the observed frequencies in d, and where, as previously, we take each pair of parameters, $\langle \Theta_i, \Theta_{i+1} \rangle$ (i odd) to be jointly Beta-distributed. Then, for each i simple algebra gives us:

$$\begin{aligned}
&\int \Theta_i{}^{a_i} (1-\Theta_i)^{b_i} \mathrm{Be}(\Theta_i \mid \alpha_i, \beta_i)\, \mathrm{d}\Theta_i \\
&\quad = \int \frac{\Theta_i{}^{a_i+\alpha_i-1}(1-\Theta_i)^{b_i+\beta_i-1}}{D(\alpha_i,\beta_i)}\, \mathrm{d}\Theta_i \\
&\quad = \frac{D(a_i+\alpha_i, b_i+\beta_i)}{D(\alpha_i,\beta_i)} \\
&\quad = \frac{\Gamma(a_i+\alpha_i)\Gamma(b_i+\beta_i)\Gamma(\alpha_i+\beta_i)}{\Gamma(a_i+\alpha_i+b_i+\beta_i)\Gamma(\alpha_i)\Gamma(\beta_i)} \\
&\quad = \frac{(a_i+\alpha_i-1)\cdots\alpha_i\Gamma(\alpha_i)(b_i+\beta_i-1)\cdots\beta_i\Gamma(\beta_i)\Gamma(\alpha_i+\beta_i)}{(a_i+\alpha_i+b_i+\beta_i-1)\cdots(\alpha_i+\beta_i)\Gamma(\alpha_i+\beta_i)\Gamma(\alpha_i)\Gamma(\beta_i)} \\
&\quad = \frac{(a_i+\alpha_i-1)\cdots\alpha_i(b_i+\beta_i-1)\cdots\beta_i}{(a_i+\alpha_i+b_i+\beta_i-1)\cdots(\alpha_i+\beta_i)}
\end{aligned}$$

And we can see from this that the predictive likelihood is given simply by iteratively calculating the probability of each instance of the sample, then updating the parameters, and repeating the process using the resulting model, as we can rearrange the factors on the top and bottom of the fraction above to be in the appropriate sequence.